

Qualitative Analysis of Farm Produce

A Project Report

Submitted by:

Pranav Gandhi (AU1940313)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

Computer Science and Engineering

at



**Ahmedabad
University**

School of Engineering and Applied Science (SEAS)

Ahmedabad, Gujarat

May, 2023

DECLARATION

I hereby declare that the project entitled “**Qualitative Analysis of Farm Produce**” submitted for the B. Tech. (**Computer Science**) degree is my original work and the project has not formed the basis for the award of any other degree, diploma, fellowship or any other similar titles.

Signature of Student

Date:

Place:

CERTIFICATE

This is to certify that the project titled “**Qualitative Analysis of Farm Produce**” is the bona fide work carried out by **Pranav Gandhi**, a student of B. Tech. (**Computer Science**) of School of Engineering and Applied Science at Ahmedabad University during the academic year 2022-2023, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in **Computer Science** and that the project has not formed the basis for the award previously of any other degree, diploma, fellowship or any other similar title.

This project was at **Sensegood Instruments Private Limited** under the supervision of the industry mentor **Dr. Sanket Patel**.

Signature of Industry Mentor

Signature of Faculty Mentor

Date:

Date:

Place:

Place:

Abstract

The aim of this project is qualitative analysis of wheat farm produce with the help of computer vision model YoloV8. This model does foreign object detection on a given image of sample of wheat farm produce which helps to determine the quality of that wheat produce. This project involves whole process from scratch. It starts with field visit to collect the samples, then collection and annotation of Data, then after training the model and building backend and frontend in order to create dashboard for the user.

Table of Contents

Declaration	i
Certificate	ii
Abstract	iii
Table of Contents	iv
List of Tables	vi
Gantt Chart	vii
1 Introduction	1
1.1 Project Definition	1
1.2 Project Objectives	1
2 Literature Survey	2
2.1 Related Work	2
2.2 Tools and Technologies	3
2.2.1 Understanding Architecture of YoloV8	3
2.2.2 Why Use YOLOv8 for this project?	4
2.2.3 Specifications of Scanner	6
3 Methodology	7
3.1 Field visit and Collection of samples	7
3.2 Data collection	7
3.3 Annotation of the data	18
3.3.1 Contours for initial annotations	18
3.3.2 Correcting the annotations	19
3.3.3 Converting annotations	36
3.4 Training the model	36
3.5 Creating User Dashboard	37
3.5.1 Backend	37
3.5.2 Frontend	37
4 Results	40
4.1 Result matrices after training	40
4.1.1 Yolov8n: Nano model (50 epochs)	40
4.1.2 Yolov8l: Large model	47
4.1.3 Model predictions on new images taken.	63
4.1.4 Results on the Dashboard	70
4.2 Project Outcomes	73
4.3 My Contributions to the Project	73

4.4	Learning Outcomes	73
4.5	Real World Applications	73
5	Conclusion	74
	Bibliography	74

List of Tables

3.2.1 Information about the data.	8
4.1.1 Class number denoting class.	41

Gantt Chart

	Week-1	Week-2	Week-3	Week-4	Week-5	Week-6	Week-7	Week-8	Week-9	Week-10	Week-11	Week-12	Week-13	Week-14	Week-15
	23-Jan-23	30-Jan-23	6-Feb-23	13-Feb-23	20-Feb-23	27-Feb-23	6-Mar-23	13-Mar-23	20-Mar-23	27-Mar-23	3-Apr-23	10-Apr-23	17-Apr-23	24-Apr-23	1-May-23
Field visit + literature survey	100%														
Data collection		100%	100%	100%	100%	100%									
Code for initial annotations							100%								
Correcting annotations								100%	100%						
Figuring out which model to use										100%					
Training the model											100%	100%			
Taking new data + testing													100%		
Creating backend for dashboard														100%	
Creating front end for the dashboard															100%

Overall completion 100%

Gantt Chart

Chapter 1

Introduction

1.1 | Project Definition

This project is aimed at creating a computer vision model that can help traders or any other buyers of farm produce to determine the quality accurately. Number of detections of foreign objects and impurities in farm produce will help them determine the quality of farm produce which will eventually help them make better pricing decisions in the auction.

1.2 | Project Objectives

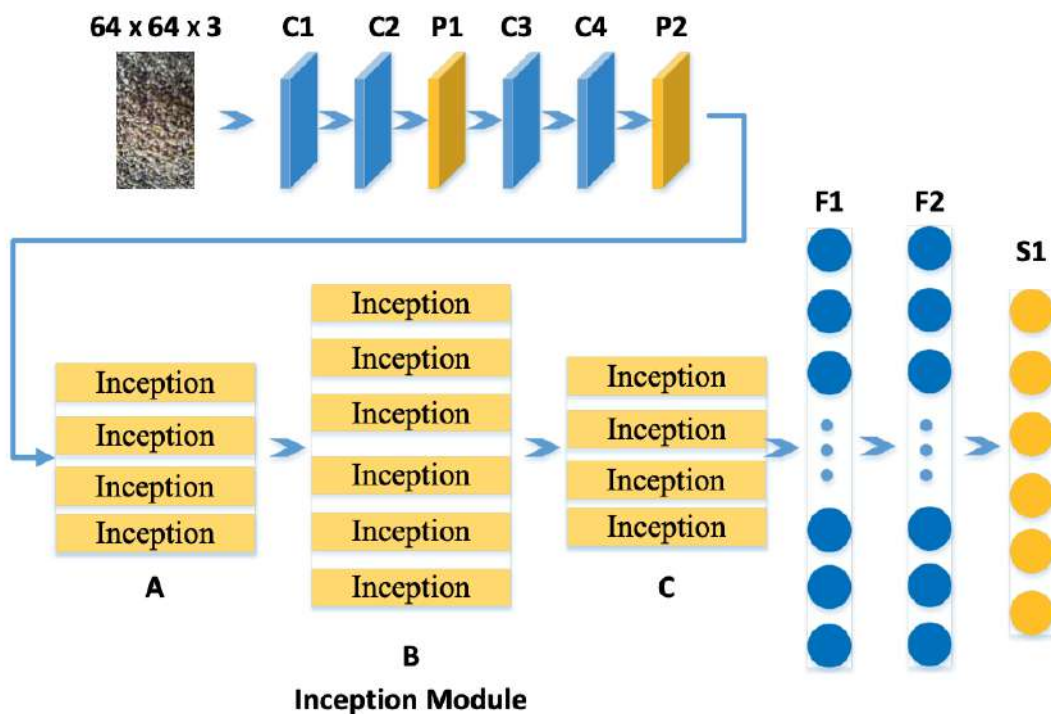
- Traders visit APMC to buy farm produce from the farmers. Traders collect some amount of farm produce from every bag that the farmer has brought. All these traders are experienced in this field and so they are able to determine the quality of that farm produce by looking at the sample collected.
- Traders visit APMC to buy farm produce from the farmers. Traders collect some amount of farm produce from every bag that the farmer has brought. All these traders are experienced in this field and so they are able to determine the quality of that farm produce by looking at the sample collected.
- One of the objective of this project is to create a model that can give number and percentages of foreign objects in the sample so that any layman without any experience in this field also can use this to determine the quality of farm produce.
- This can help traders scale their business because now they can send their employees (who are layman) and can get the analysis of farm produce using our model by just sitting at his office. Trader can participate in multiple auctions happening at different places at the same time by getting analysis of these samples using our model.
- A farmer cannot show his subjective analysis as proof for quality of his farm produce. For this, our model can be very useful because a farmer can show the analysis as an objective analysis by showcasing it in a form of certificate or report before the auction. The use case of this is same for farmer or trader who is exporting wheat farm produce. The quality report of analysis by our model can be crucial for winning the trust of client who is importing from the trader.

Chapter 2

Literature Survey

2.1 | Related Work

- There is not much research done on foreign object detections in wheat using computer vision models. However, I was able to find one paper from IEEE access journal [1].
- The dataset was taken in 2019 from the Xiaotangshan, Beijing, China. 6,000 images of wheat in the grain elevators was collected during operation using industrial camera. Camera had 16 mm focal length lens. Resolution was 2,000,000 pixels.
- This paper proposes an improved version of CNN called WheNet to recognize and detect impurities/foreign objects in wheat.



Structure of the WheNet CNN

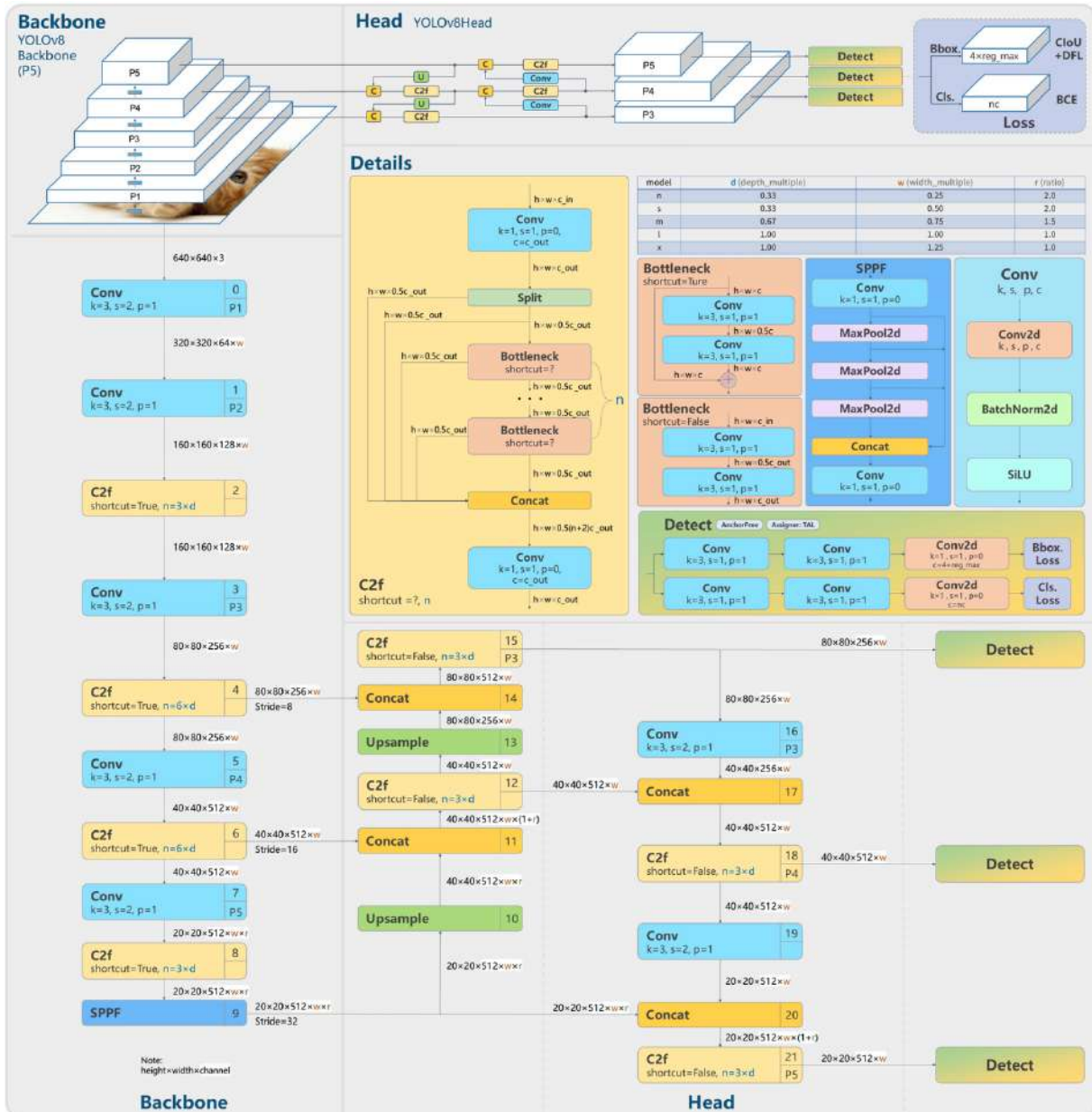
- Few problems related to using this paper: There was no code and no dataset provided by the authors of this paper and even if the dataset was available then also it would

be not useful to us because it is farm produce from China. There are differences in wheat grains and also in types of impurities found in wheat farm produce in India compared to that of China. Also the use case for which this paper is about does not seem to align with our use case because data include images of wheat grains in lot of Clumped manner where it is only recognizing if there is any impurity found on the top whereas in our case we want to detect and classify each and every object of our sample to give analysis.

2.2 | Tools and Technologies

2.2.1 | Understanding Architecture of YoloV8

- The architecture of this model is build on previous versions of YOLO models. YOLOv8 utilizes CNN which is divided in two parts: Head and Backbone.
- Backbone is a modification of architecture named CSP Darknet53. It has 53 Convolutional layers. It uses cross stage partial connections to make the information flow better [2].
- The head has many convolutional layers followed by a series of fully connected layers. Class probabilities, bounding box and objectness score for the objects detected in the image are due to these layers. Self-attention mechanism is used in the head part which makes the model able to focus on various parts of the image and give importance to different features accordingly.
- The model utilizes a feature pyramid network which gives it ability to detect small as well as large objects in the same image.

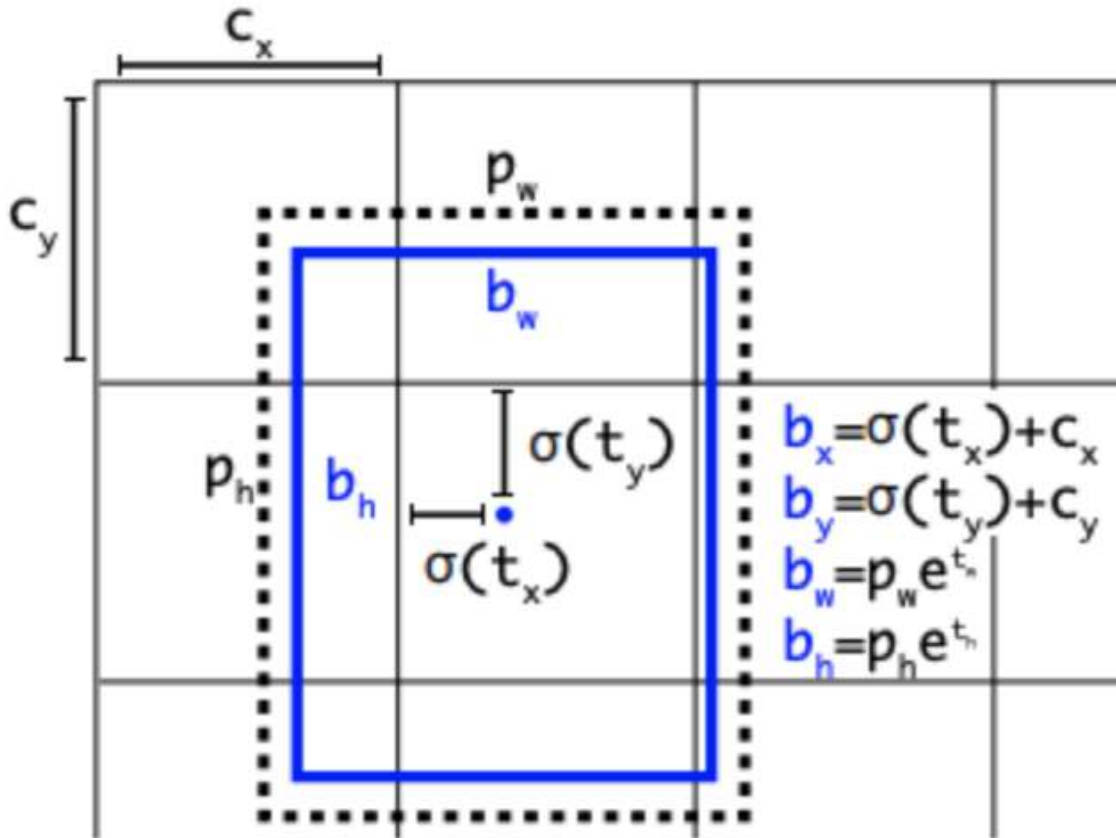


Architecture of YOLOv8 [3]

2.2.2 | Why Use YOLOv8 for this project?

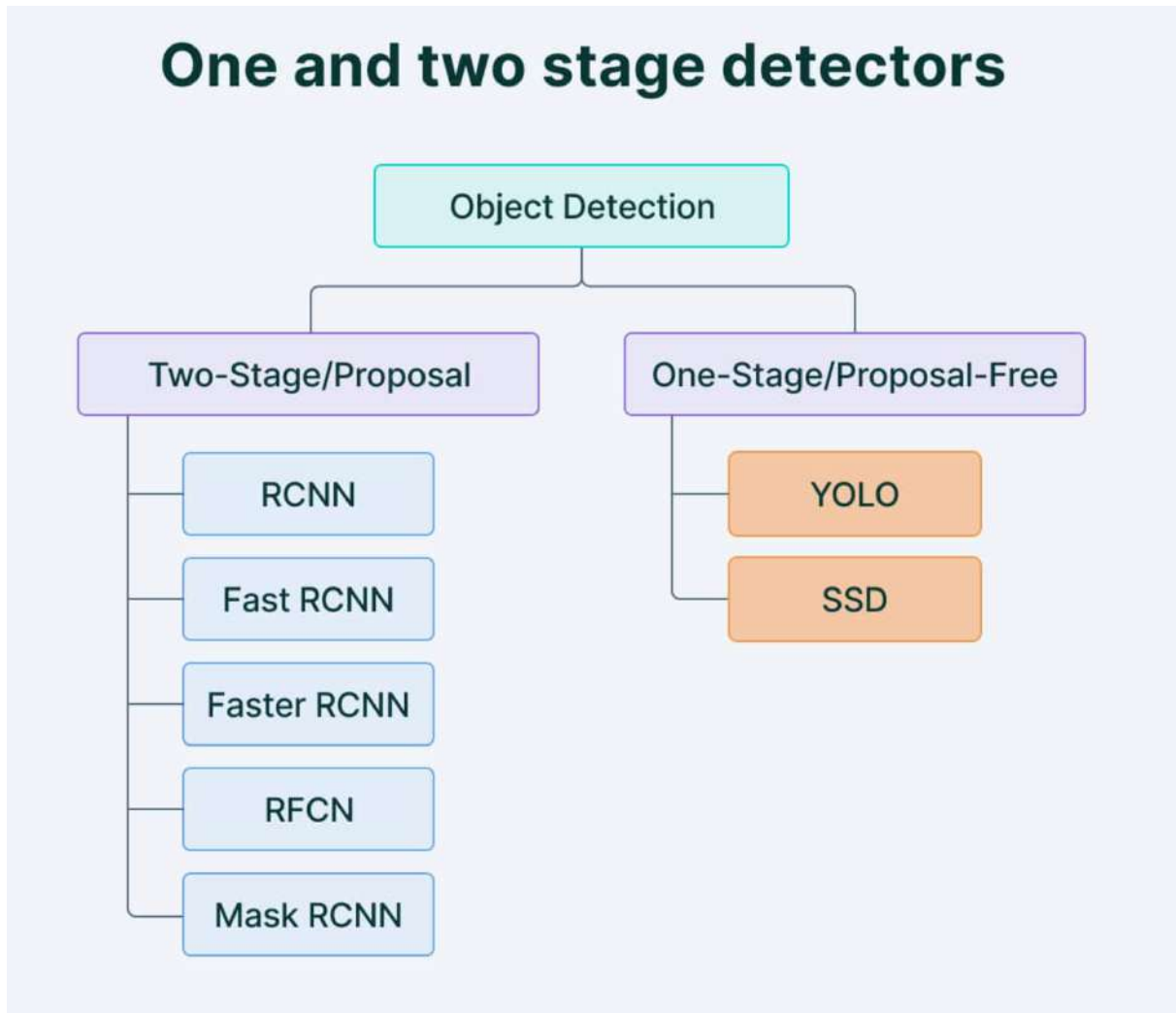
- Generally, models use anchor boxes to make bounding boxes prediction. Most of the state of the art models use anchor boxes as a prior to predict and then localize multiple objects in a image. The process looks like this:- First it makes candidate anchor boxes, then for every box some offset value is predicted, then based on the gorund truth the loss function is calculated, then calculate probability based on how much the overlap is between the real object and offset box and then finally the prediction is factored to the loss function.
- In our dataset, the shapes of different class of objects are very different from each other. Also objects of Class bran and stones have very irregular shape. For this reason, custom tuning the anchor boxes would be a necessity. Probably this would require me to custom tune anchor boxes for each class separately.

- YOLOv8 best suits for our problem because it is an anchor-free model. YOLOv8 does anchor free detection. Instead of calculating offset from a known anchor box, it predicts directly the center of an object.
- Anchor free detection speeds up Non-maximum suppression (NMS) because number of box predictions is reduced.



Anchor box in YOLO [3]

- Single-shot object detection: Yolo looks at the image only once, in order to make predictions about presence and location of the objects. This makes the process computationally efficient compared to two shot object detection models.



Stage Detectors [4]

2.2.3 | Specifications of Scanner

Scanner used for collecting the data was CanoScan LiDE 300 (color image scanner). Resolution was set to 400dpi.

Chapter 3

Methodology

3.1 | Field visit and Collection of samples

- I visited APMC mandi (Jamnagar, Gujrat) in order to understand how traders buy farm produce in an auction and also to learn about the types of impurities/foreign objects found in wheat produce.
- Samples were collected from APMC mandi Jamnagar and then I also visited a local factory where foreign objects/impurities are separated from the wheat farm produce and then wheat grains are packed to sell further to vendors and other local traders. From here, I collected the samples of foreign objects separately. [Find my field visit videos here](#)

3.2 | Data collection

- Data was collected as scanned images of farm produce sample put on a scanner with blue sheet for background.
- The scans were taken with image resolution 400dpi. Dimension of each image is 3400X4677. Images are taken in JPEG format
- Wheat farm produce has 6 types of impurities/foreign objects:
 - Stones
 - Bran
 - Wheat grains with bran
 - Small wheat grains
 - Stalk
 - Broken wheat grains
- In total there are 7 classes: 6 is of the above mentioned impurities and remaining one is of Big wheat grain (which is the final product we care about).
- Data is taken in different combinations as well as separately:
 - Only Big wheat grains

- Only Stones
 - Only Small wheat grains
 - Only Broken wheat grains
 - Only Stalk
 - Only Bran
 - Only Wheat grains with Bran
 - Combination1: Stones and Big wheat grains
 - Combination2: Stones and Grains(both big and small)
 - Combination3: Big grains + Wheat grains with brans + stalk + stones
- The following table, Table 3.2.1, shows number of images taken for each type of above mentioned combinations.

Table 3.2.1: Information about the data.

Type	Number of Images
Only Big wheat grains	83
Only Stones	40
Only Small wheat grains	35
Only Broken wheat grains	30
Only Stalk	22
Only Bran	20
Only wheat grains with bran	30
Combination1	37
Combination2	31
Combination3	17
Total	345

- Also images are taken for each of these classes where objects are clumped with each other. Below are some images.



BigWheat.jpg



BigWheatClumped12.jpg



Stones



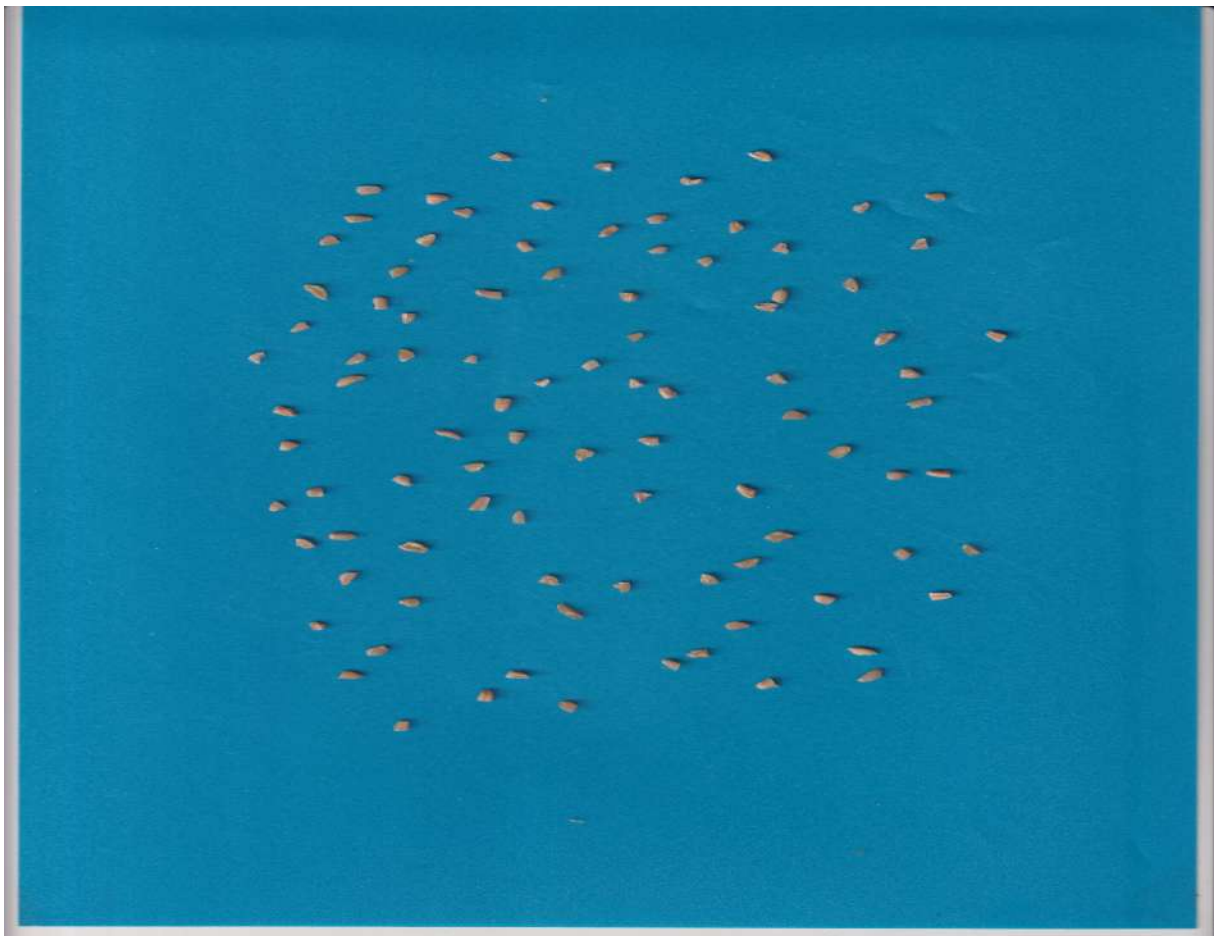
Clumped Stones



Small wheat grains



Clumped Small wheat grains



Broken wheat grains



Clumped Broken wheat grains



Stalk



Bran



Wheat grains with bran



Clumped Wheat grains with Bran



Stones and Big wheat grains



Clumped Stones and Big wheat grains



Stones and Big and Small wheat grains



Mix



Mix Clumped

3.3 | Annotation of the data

3.3.1 | Contours for initial annotations

- There are total 345 images in our dataset and in each image there are many objects. To annotate all these objects from scratch would have taken a lot of time. So I used opencv contours in order to get the initial annotations.
- Using hsv color slider, the values of hsv to filter out the background was found. The code for this slider is in colorslider.py file which is in the drive link given at the end of this section.
- After figuring out the hsv values of the background, lower and upper hsv values is used to set a threshold. This threshold is given to cv2.findContours() in order to get the contours. The part of my code doing this is in the below image.
- Annotations are stored in COCO format as a json file. Part of my code doing this is shown in the below image.

```

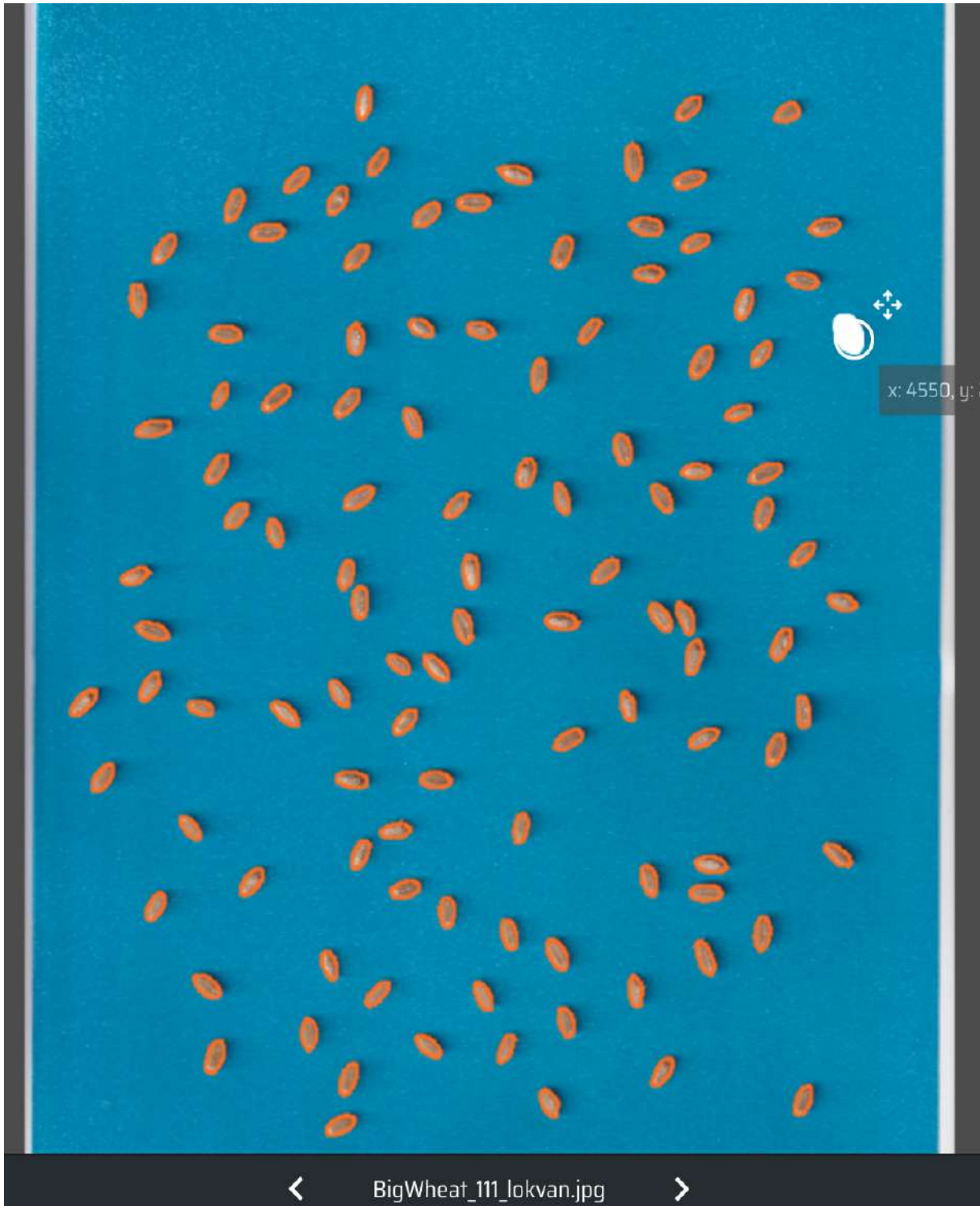
86     # update polygons json
87     polygons.update({
88         i: {
89             "grain_file": saveFilename,
90             "source_file": filepath,
91             "width": dst.shape[0],
92             "height": dst.shape[1],
93             "polygon": cnt2.flatten().reshape((-1, 2)).tolist()
94         }
95     })
96
97     anns.append({
98         "id": annid,
99         "iscrowd": 0,
100        "image_id": imgid,
101        "category_id": 1,
102        "segmentation": [cnt.flatten().tolist()],
103        "bbox": [x,y,w,h],
104        "area": w * h
105    })
106    annid+=1
107    i += 1
108    cv2.imwrite("grain_images/" + saveFilename, dst)
109
110    imgid+=1
111    with open("grain_polys.json", "w") as f:
112        json.dump(polygons, f)
113
114    with open( "wheat-grains-coco.json", "w") as f:
115        json.dump({
116            "info": {
117                "description": "ccpd_green"
118            },
119            "images": imgs,
120            "annotations": anns,
121            "categories": cats
122        }, f)
  
```

Storing annotations in COCO format

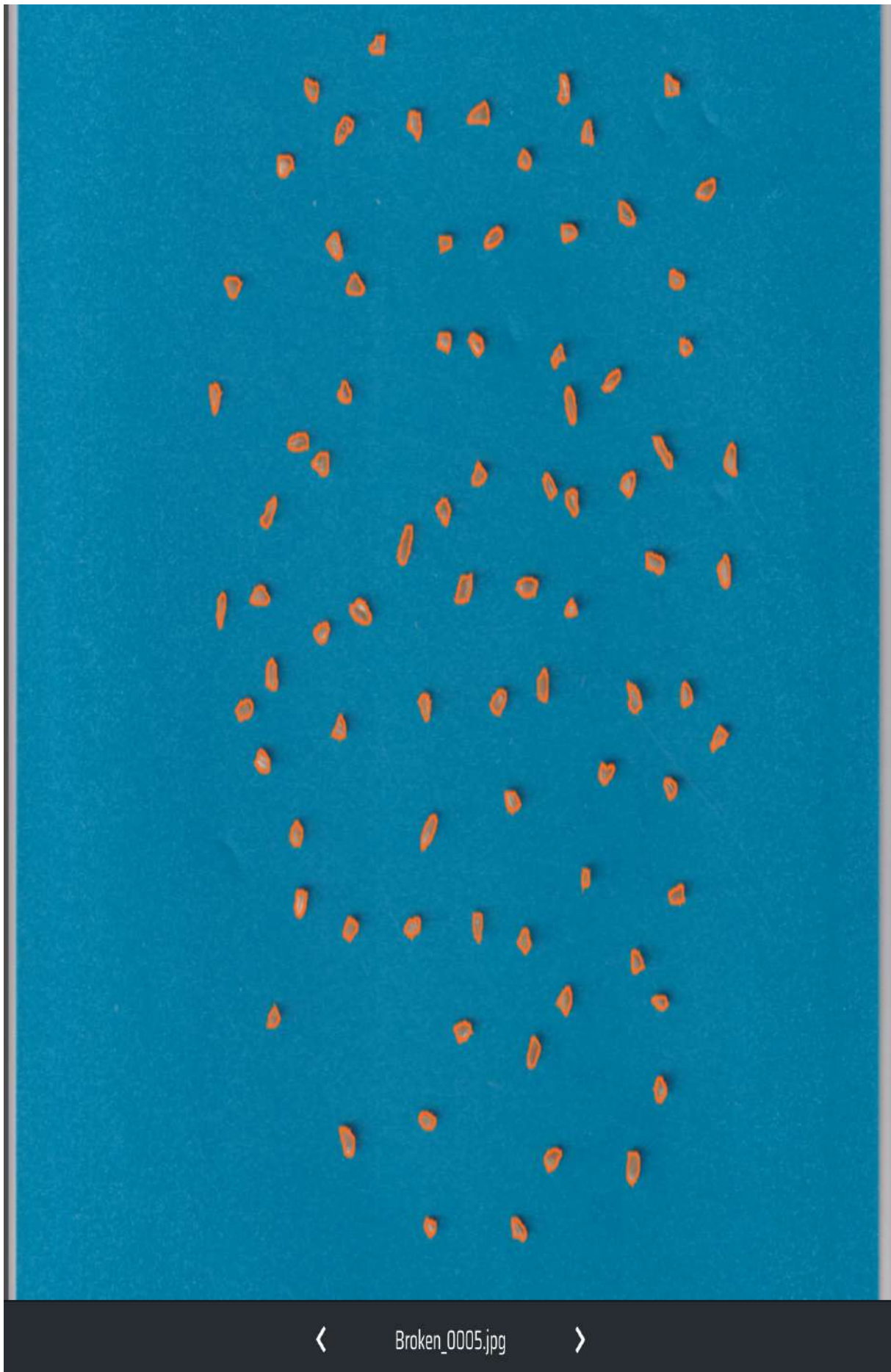
[Find my code for these annotations here.](#)

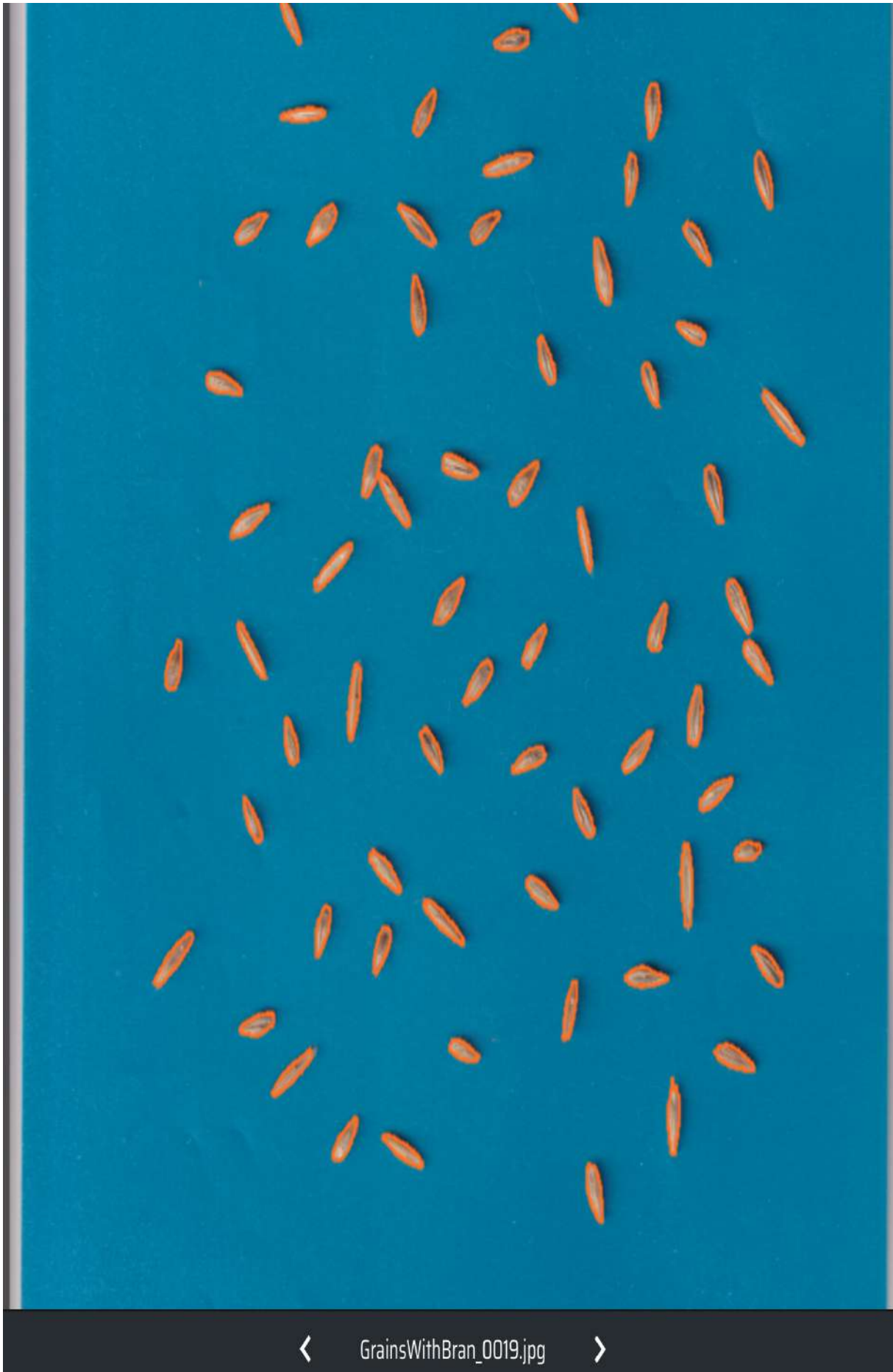
3.3.2 | Correcting the annotations

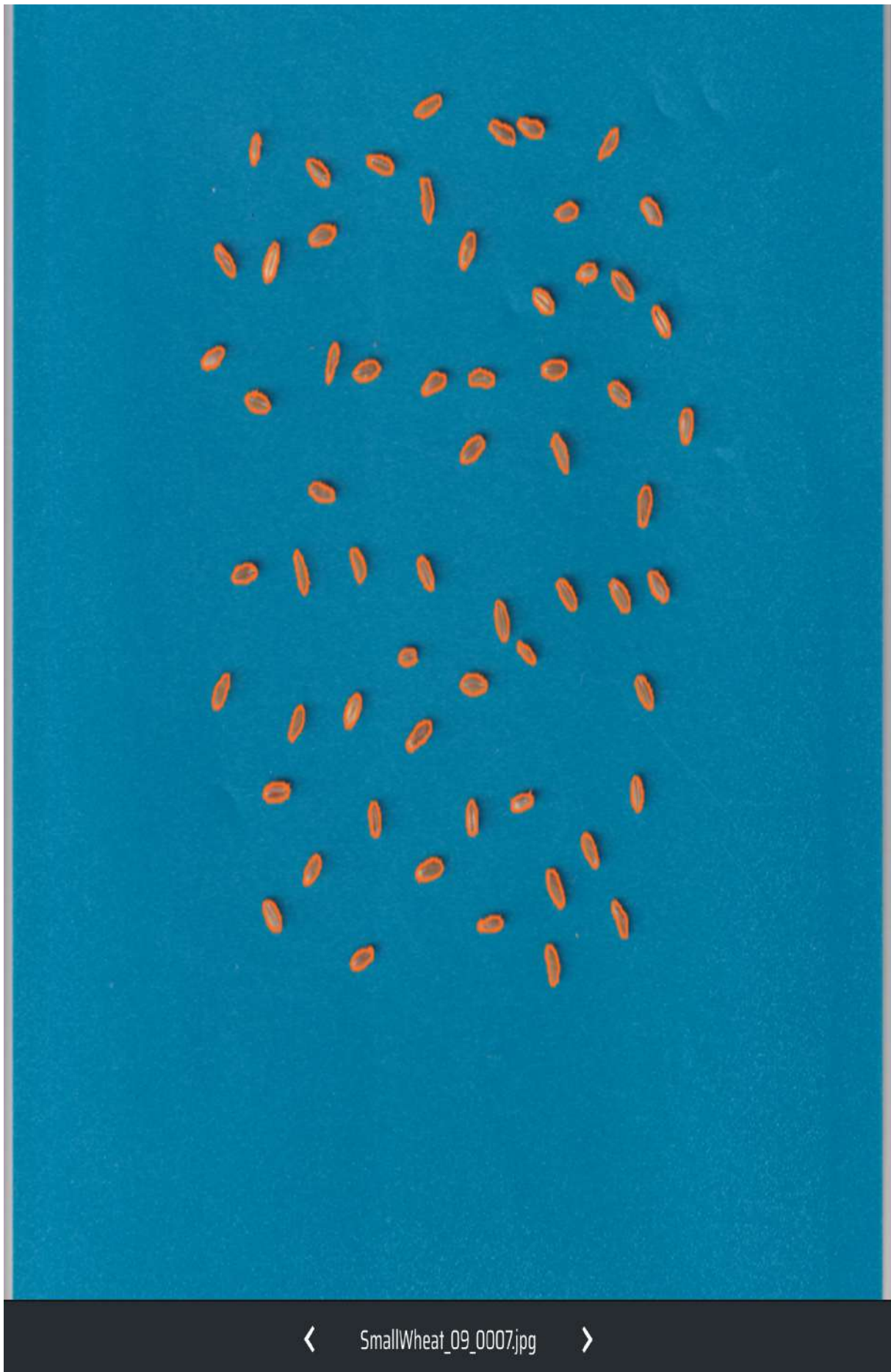
- The contours worked well in the objects which are not clumped as shown in the below images.





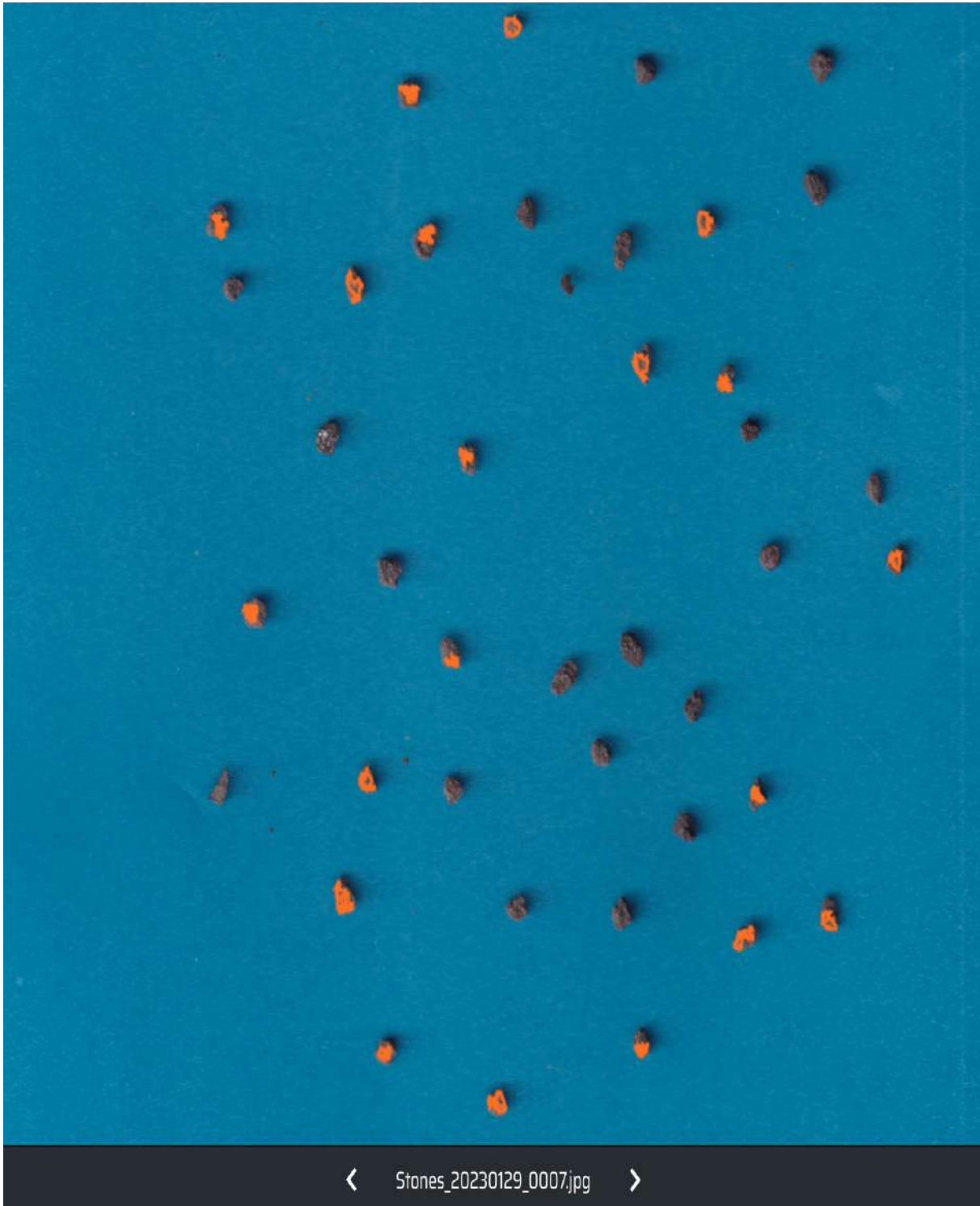








- However, contours didn't work for almost all stones, few stalks and some of Wheat brans. Major problem of this method is that it does not work for clumped objects. If two or more objects are clumped, then contour is formed around all of them and which makes all of them as one object. Some of the errors are shown in below images.









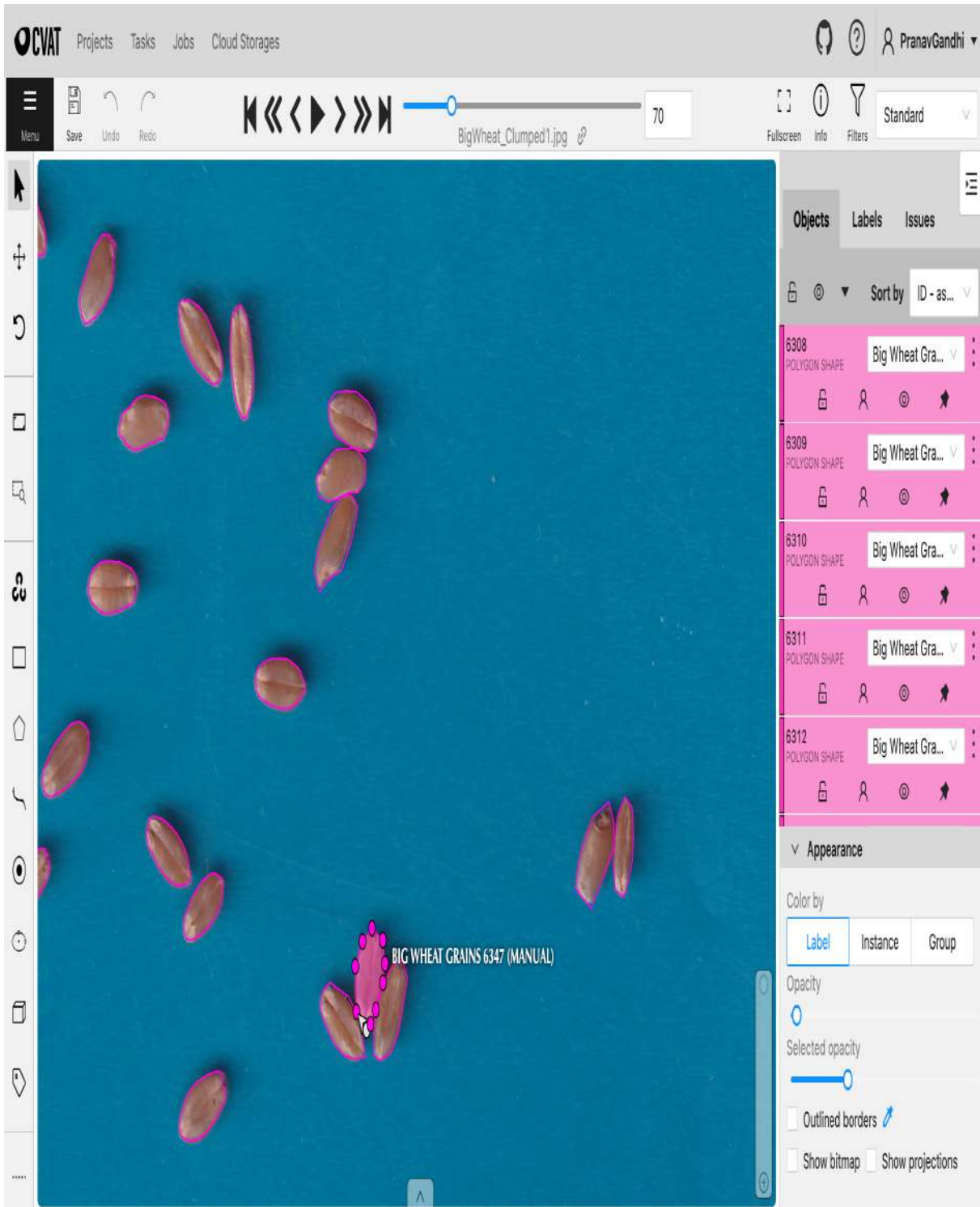


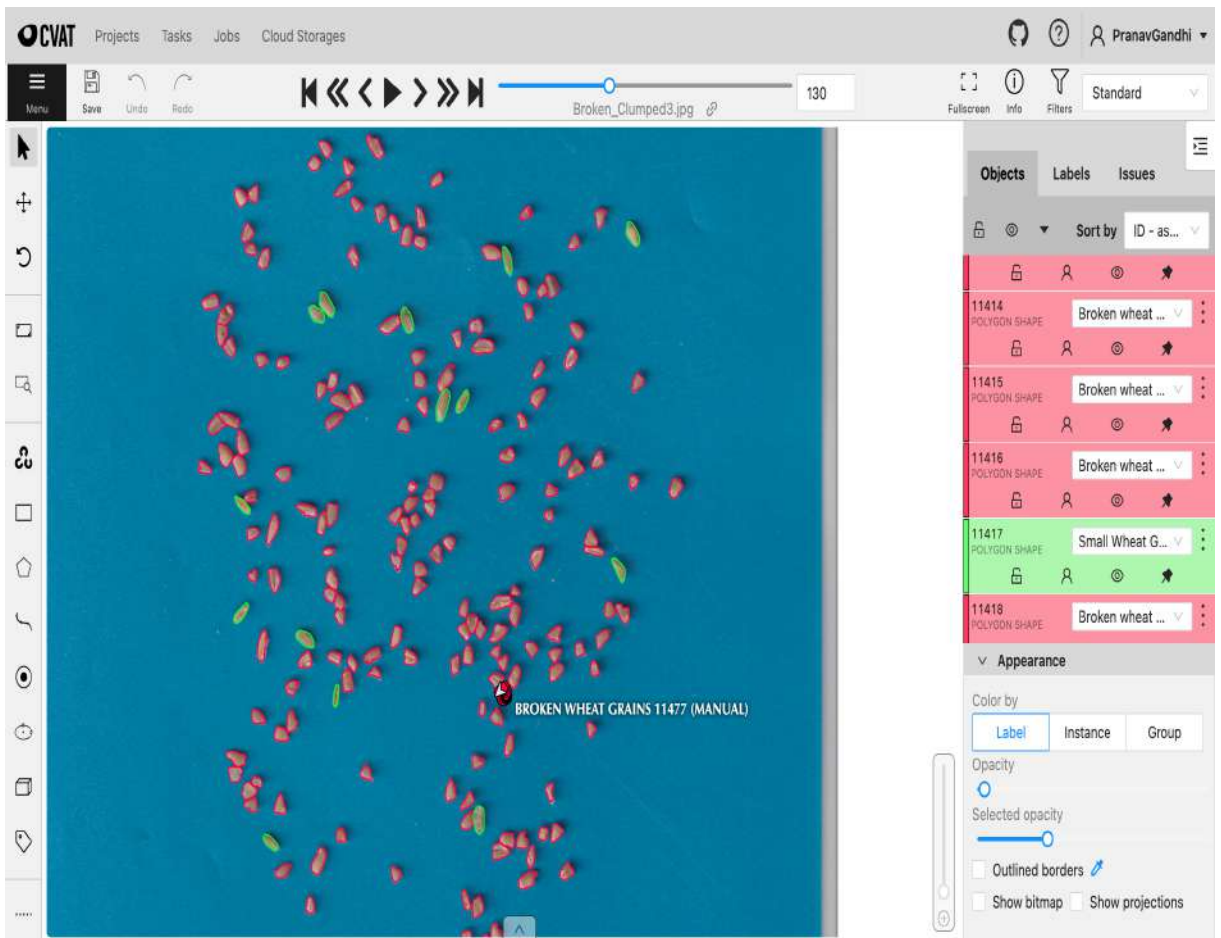




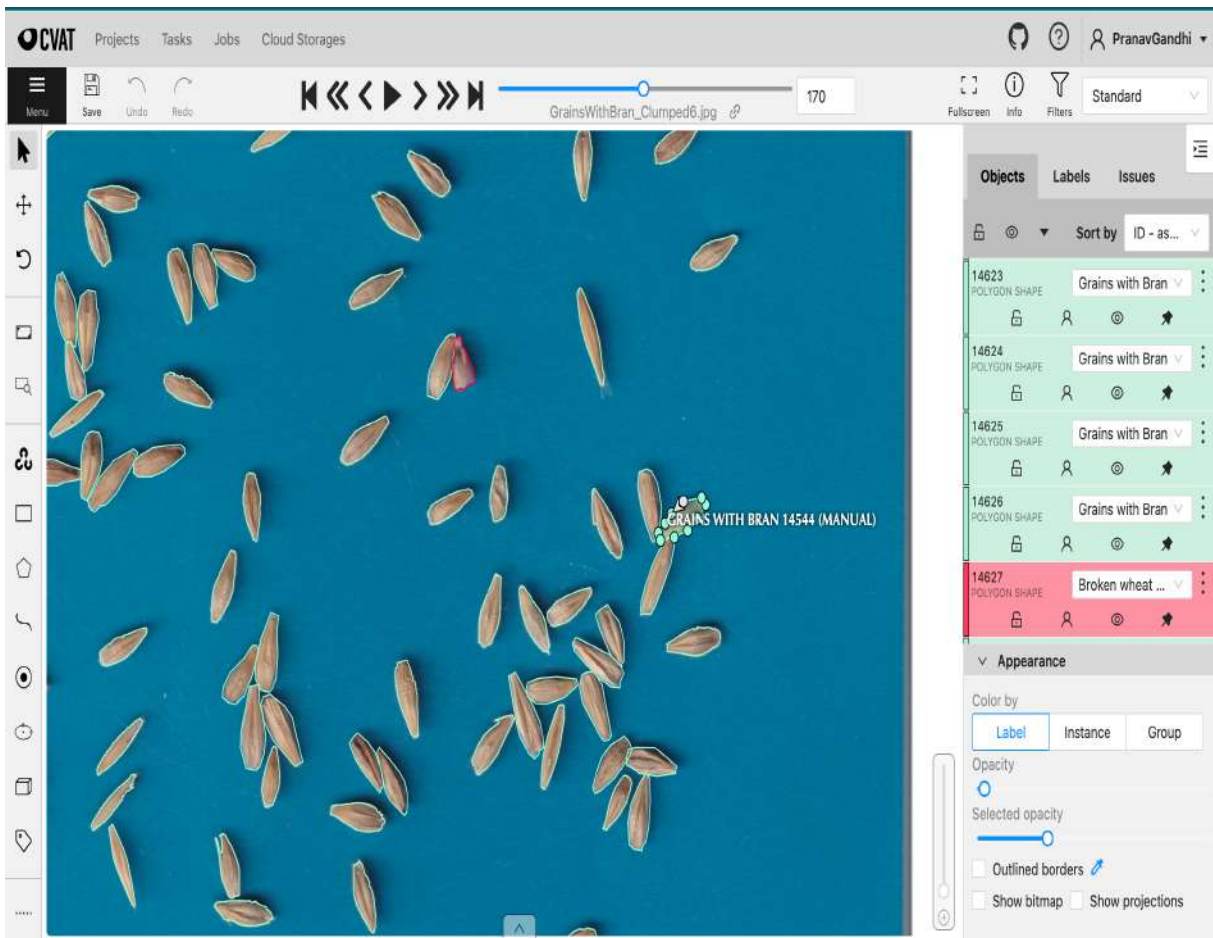
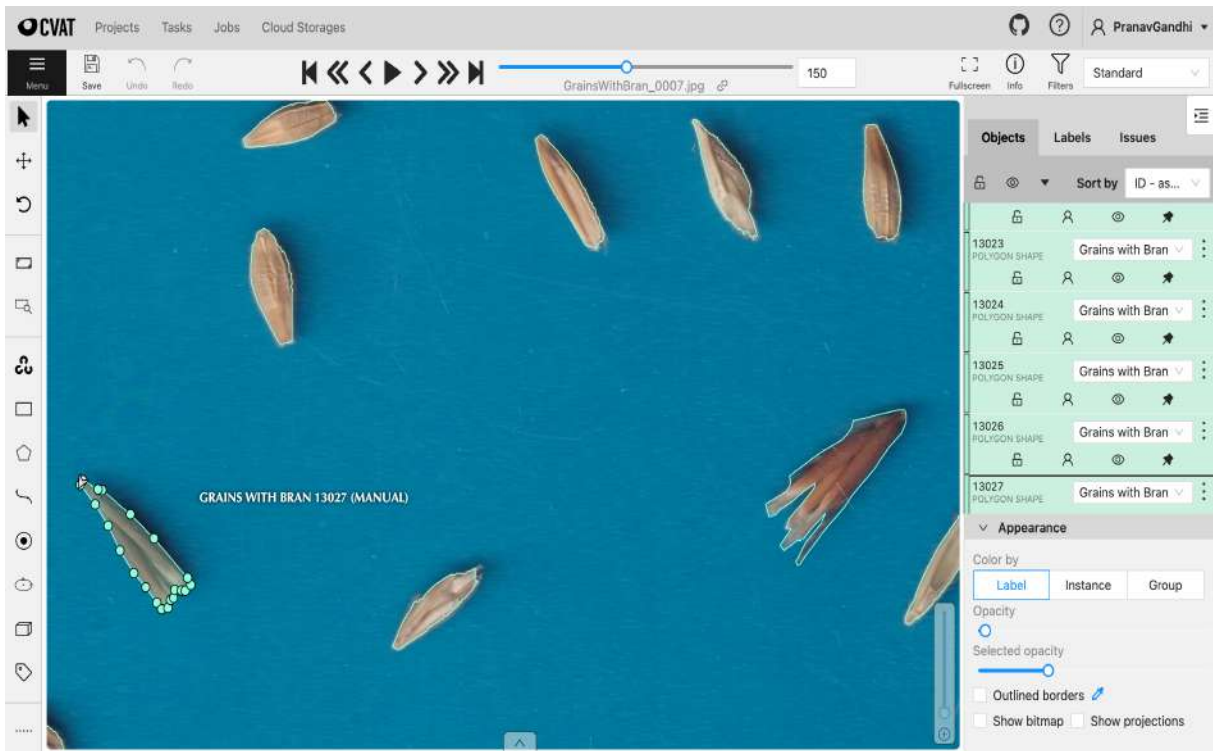
< GrainsWithBran_0016.jpg >

- I had to manually delete all these errors in the annotations and then draw polygon annotations around these objects. Computer vision annotation tool was used to correct these annotations. Examples of some of the corrections are shown in below images.





The screenshot displays the CVAT (Computer Vision Annotation Tool) interface. At the top, the navigation bar includes 'CVAT', 'Projects', 'Tasks', 'Jobs', and 'Cloud Storages', along with a user profile for 'PranavGandhi'. The toolbar below features navigation icons (back, forward, home), 'Save', 'Undo', and 'Redo' buttons, a zoom slider set to 142%, and 'Fullscreen', 'Info', and 'Filters' options. The main canvas shows a blue-tinted image of wheat grains with several green polygons highlighting stones. A 'Draw new polygon' dialog is open, showing a 'Label' dropdown set to 'Small Wheat Grains' and a 'Number of points' input field. The right sidebar contains an 'Objects' panel with a 'Sort by' dropdown set to 'ID - as...', a list of objects (12820-12824) labeled 'Stones' with 'POLYGON SHAPE' type, and an 'Appearance' panel with 'Color by' options (Label, Instance, Group), an 'Opacity' slider, and checkboxes for 'Outlined borders', 'Show bitmap', and 'Show projections'. A specific object is labeled 'STONES 12847 (MANUAL)'.



[These corrected annotations can be found here in COCO format.](#)

3.3.3 | Converting annotations

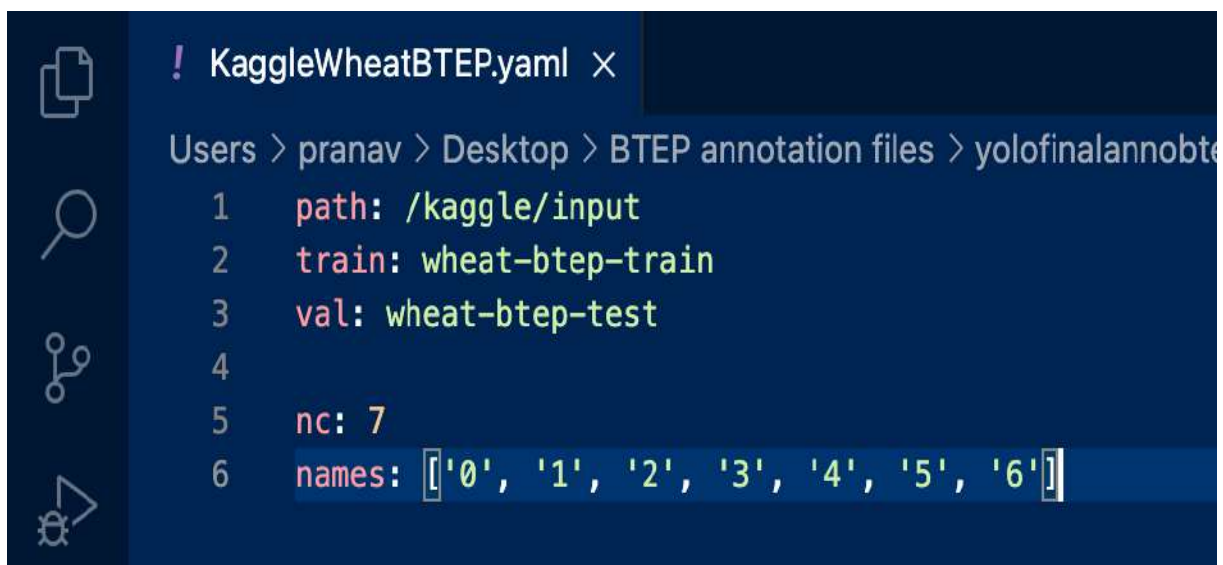
- These corrected annotations are then exported in COCO json format from cvat. However, training a YOLO model requires annotations to be in YOLO format.
- Conversion to Yolo format was done using code provided in Github of Ultralytics (which is the company who created YOLOv8). [The code can be found here.](#)

3.4 | Training the model

- Splitting of dataset: 314 images for training and 31 images for testing. Images for testing is taken from each of the types mentioned in Table 3.2.1. [Dataset can be found here.](#)
- There are different sizes of YOLOv8 models. But for this project, models trained:
 - YOLOv8n: nano model (3.2 M parameters) upto 50 epochs
 - YOLOv8l: large model (43.7 M parameters) upto 120 epochs
- Training was done in my kaggle account notebook using P100 GPU provided by kaggle. CLI command was used to train as shown below.

```
!yolo task=detect mode=train model=yolov8l.pt
data="/kaggle/input/btep-data-yaml/KaggleWheatBTEP.yaml"
epochs=40 imgsz=2048 batch=1 device=0
```

- KaggleWheatBTEP.yaml file given is for specifying paths of the data files, names of classes and number of classes.



```
! KaggleWheatBTEP.yaml x
Users > pranav > Desktop > BTEP annotation files > yolofinalannobte
1 path: /kaggle/input
2 train: wheat-btep-train
3 val: wheat-btep-test
4
5 nc: 7
6 names: ['0', '1', '2', '3', '4', '5', '6']
```

kaggleWheatBTEP.yaml file

- Outputs and performance of these trained models are shown in Results (Chapter 4). [The yolov8l trained model is here.](#)

3.5 | Creating User Dashboard

3.5.1 | Backend

- Backend is created using FastApi in python.
- First the image is taken from the frontend, then it is read and bytes are stored in numpy array. Then it is given to the model using predict(). Then the results are encoded and stored in the new image variable which is passed to frontend. Snippet from my code is shown in below image.

```

29 @app.post("/uploadimage/")
30 async def create_upload_file(file: UploadFile):
31     print('Got File')
32     image = np.array(Image.open(BytesIO(await file.read())).convert('RGB'))
33     print(image.shape)
34     # new_image = image.transpose()
35     image = image[:, :, ::-1]
36     results = model.predict(image, save=False, max_det=1000, imgsiz=2048, hide_labels=True)
37     new_image = base64.b64encode(cv2.imencode('.jpg', results[0].plot())[1]).decode("utf-8")
38     return {
39         'predictions': results[0].boxes.cls.tolist(),
40         'image': new_image
41     }
42
43 if __name__ == '__main__':
44
45     uvicorn.run(app=app, host='127.0.0.1', port=8080)
  
```

3.5.2 | Frontend

- Frontend is created using Streamlit in python.
- Data from backend is stored in json format in variable named data. From that, class predictions of every object detected are stored in another variable called classarray. After that count of objects of each class is taken and stored as a value in dictionary variable d. This dictionary is then converted into dataframe and displayed as a table in the frontend.


```
ans = requests.post(url = "http://127.0.0.1:8080/uploadimage/", files={"file": file.getvalue()})
data = ans.json()
class_array=data['predictions']
print(class_array)
d = Counter(class_array)
d = dict(d)
```

- Image data from backend is then decoded using `opencv imdecode()`. The snippet of my code is shown below.

```
file_bytes = np.frombuffer(base64.b64decode(data['image']), dtype=np.uint8,)
img = cv2.imdecode(file_bytes, flags=1)
color_coverted=cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
# pil_image=Image.fromarray(color_coverted)
# predicted_image = Image.fromarray(file_bytes, 'RGB')
streamlit.image(color_coverted)
```

[Frontend Backend code files are here.](#)

- This is how the Dashboard looks like:



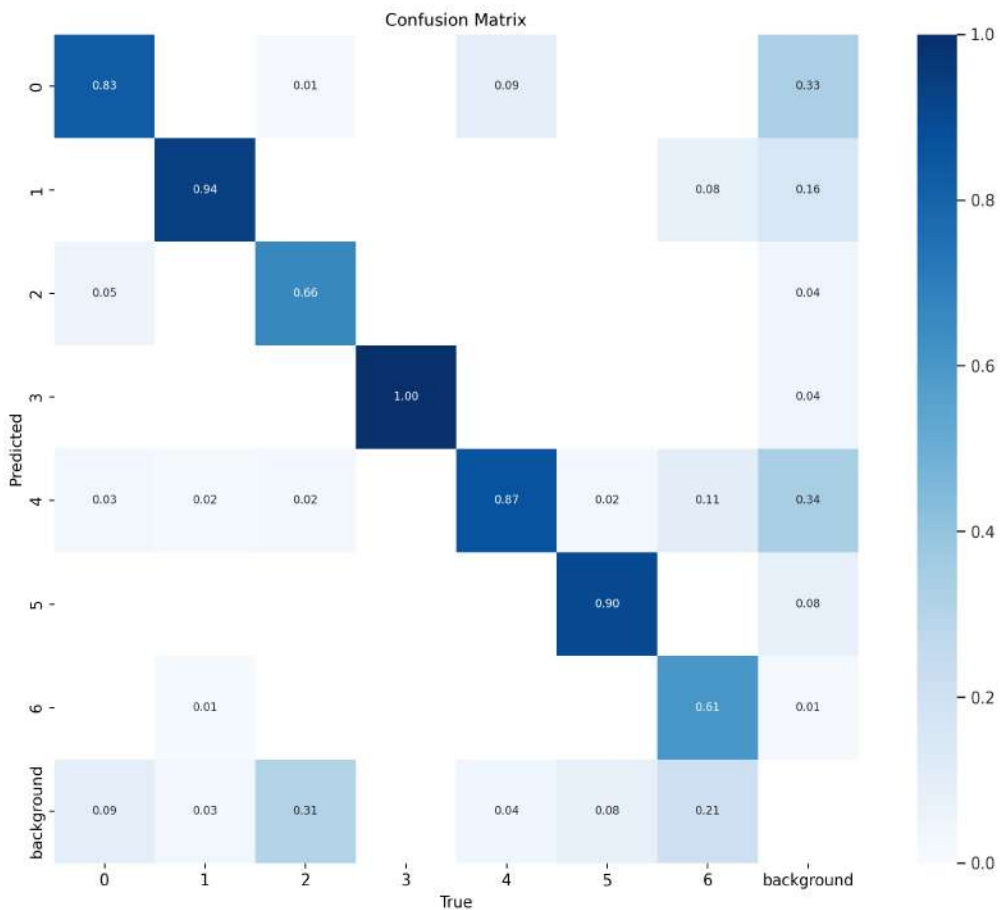
Chapter 4

Results

4.1 | Result matrices after training

4.1.1 | Yolov8n: Nano model (50 epochs)

- Below is confusion matrix which is showing the true class vs the predicted class.



Yolov8 Nano model confusion matrix

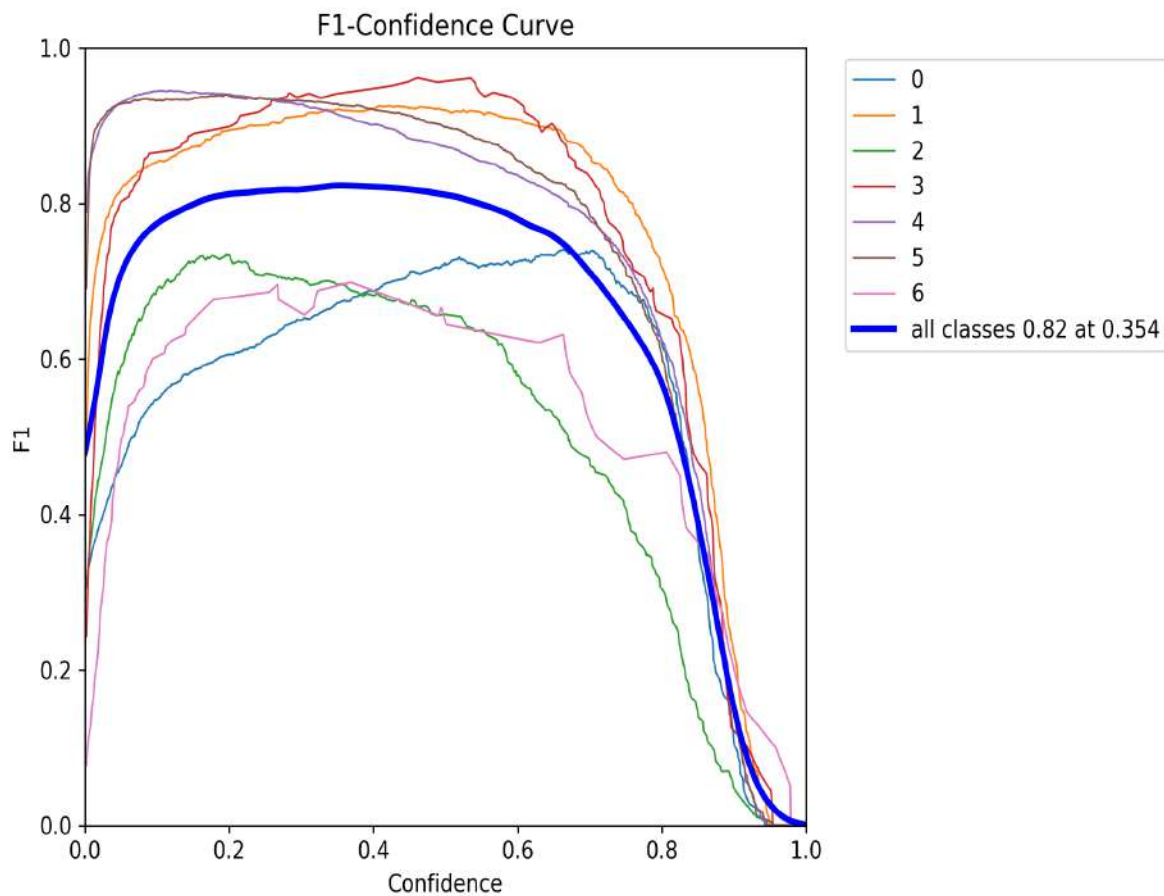
- From the above image, it shows the performance of the model is very poor.

- The following table, Table 4.1.1, shows class number assigned to denote each type of class.

Table 4.1.1: Class number denoting class.

Class number	Class
0	Small Wheat Grains
1	Grains with bran
2	Broken wheat grains
3	Bran
4	Big wheat grains
5	Stones
6	Stalk

- This model is struggling to predict classes 0 and 2. This means it is not able to predict and learn objects which are very small, which in our case is small wheat grains and broken wheat grains.
- This model is also struggling to predict class 6 which is stalk. The reason might be because some of the stalk objects are very large in size and some are very small.
- Below is the F1 curve.



F1 confidence curve

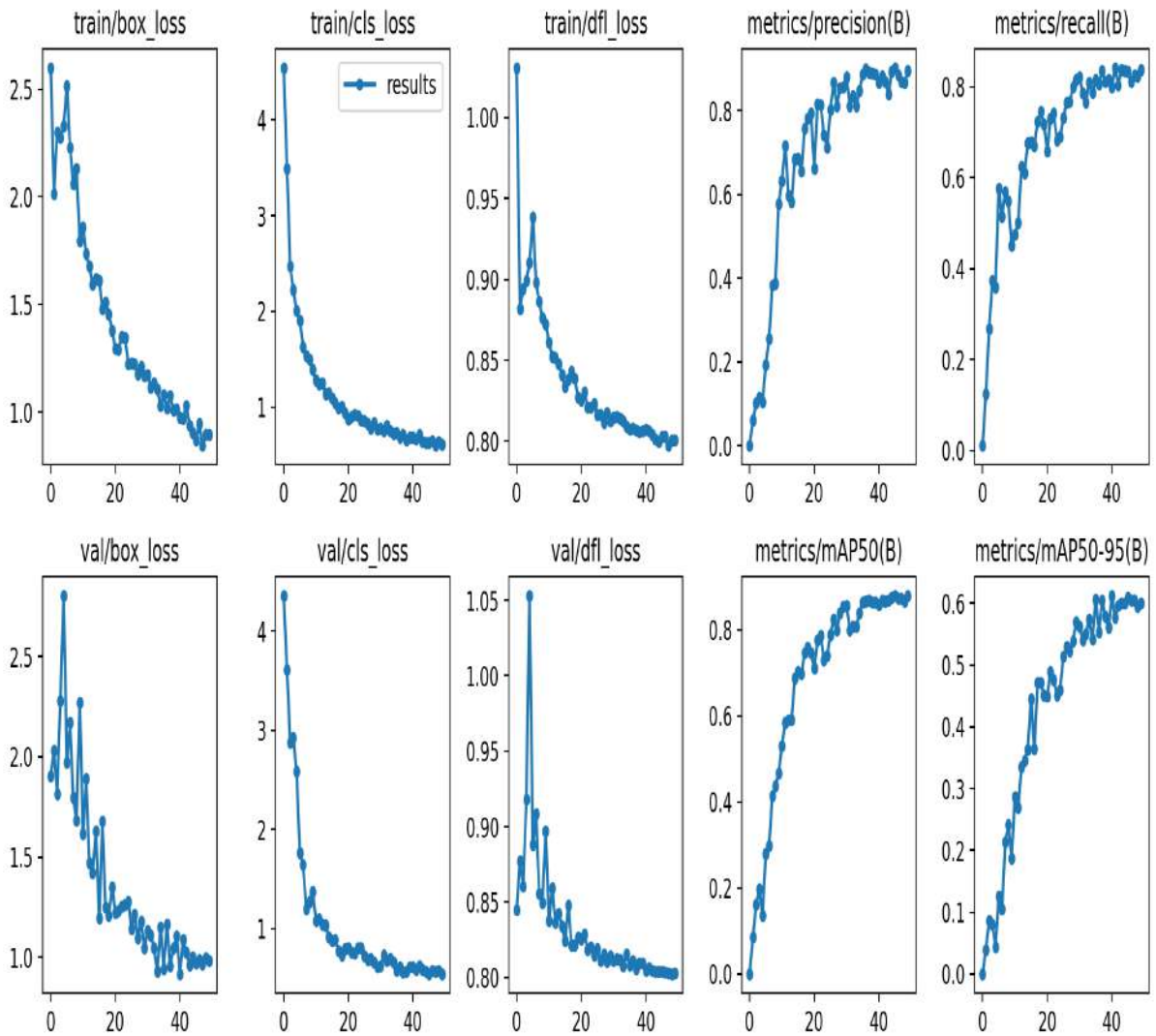
$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

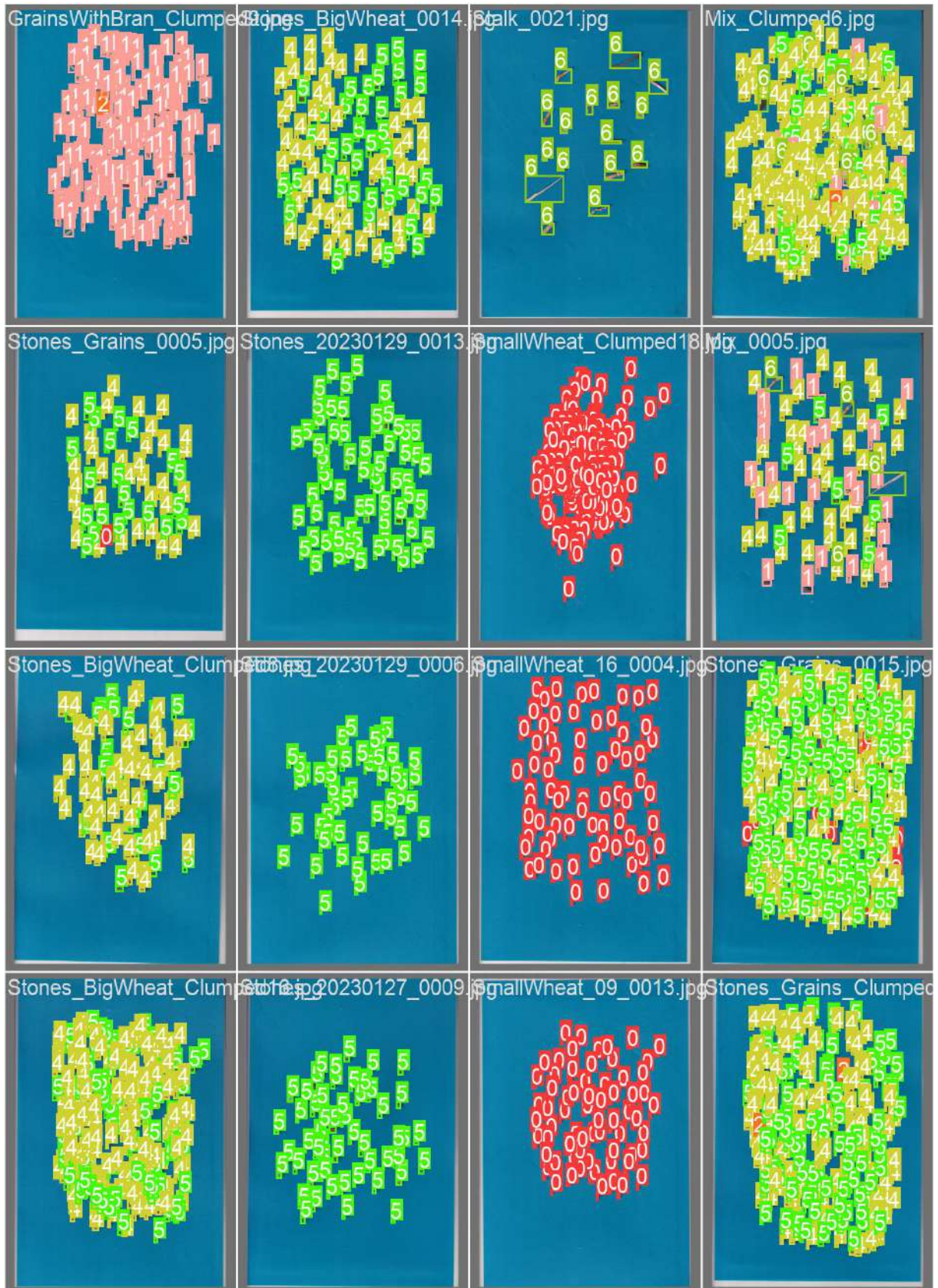
Explaining F1 [5]

- As we can see, even at 0.34 confidence, maximum f1 is 0.82 which is low.
- Below is results shown.

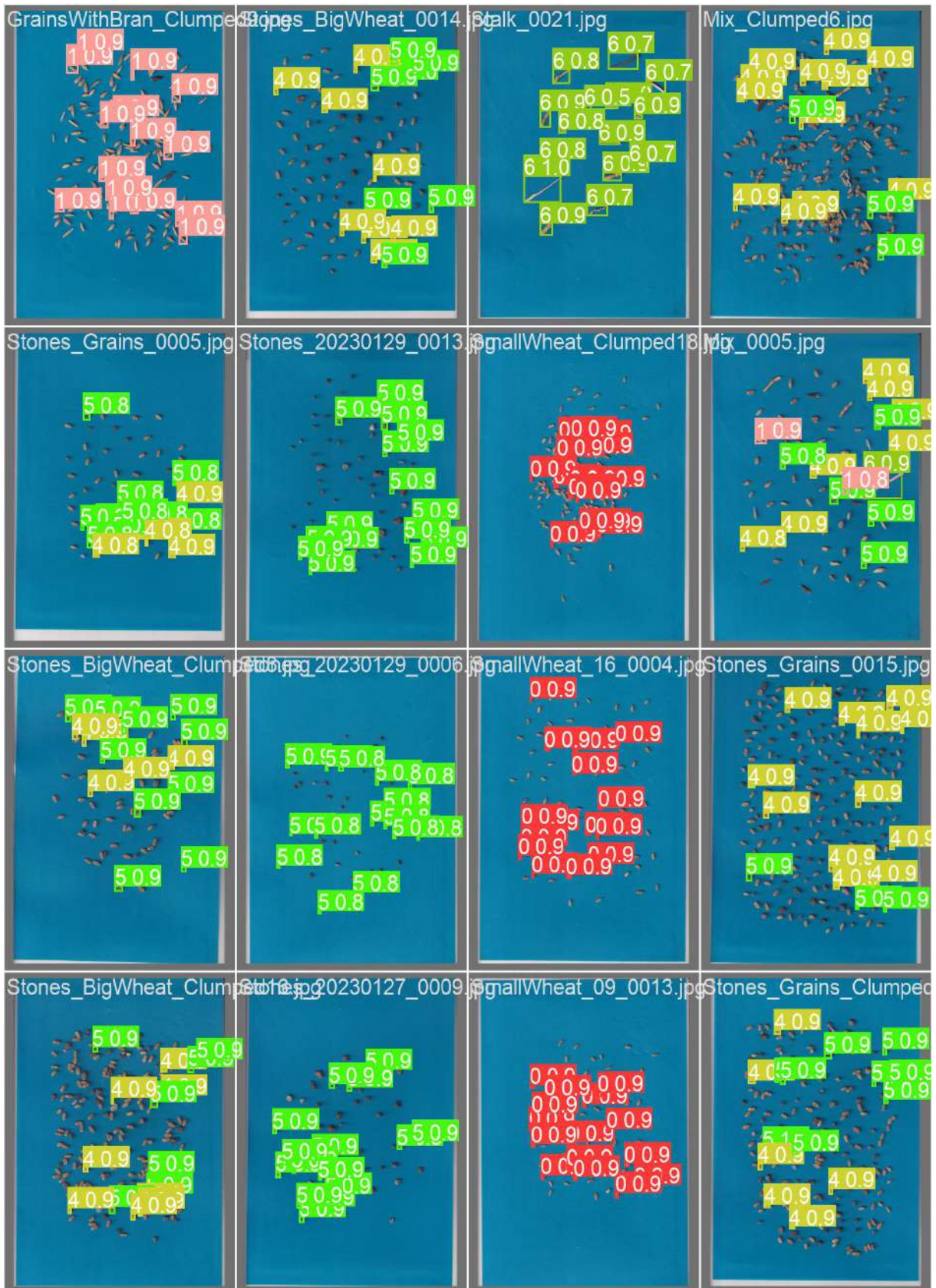


Results

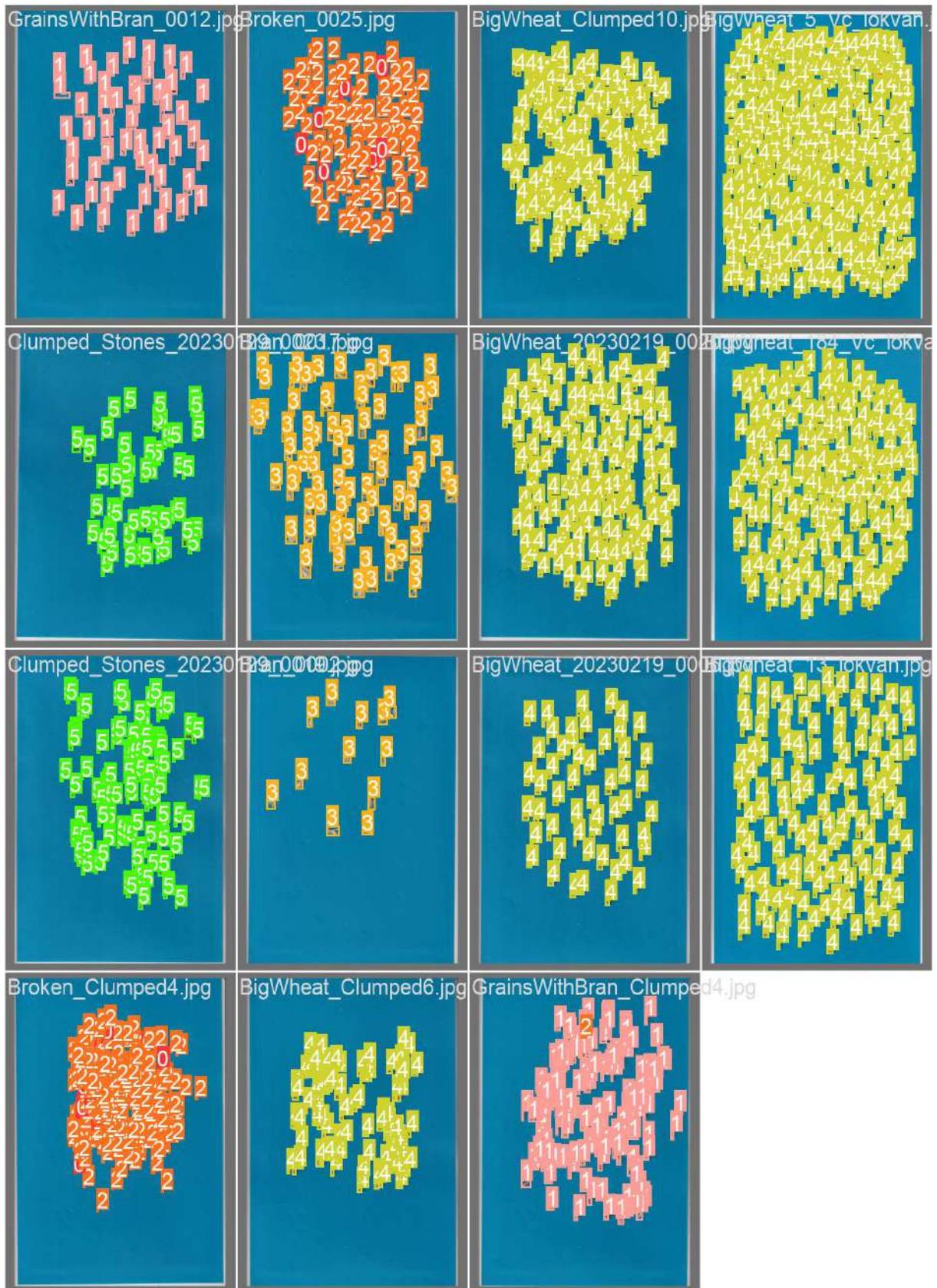
- Below images are validation batches of what we have labeled and what the model has predicted. This will help us see how the model has performed on test images.



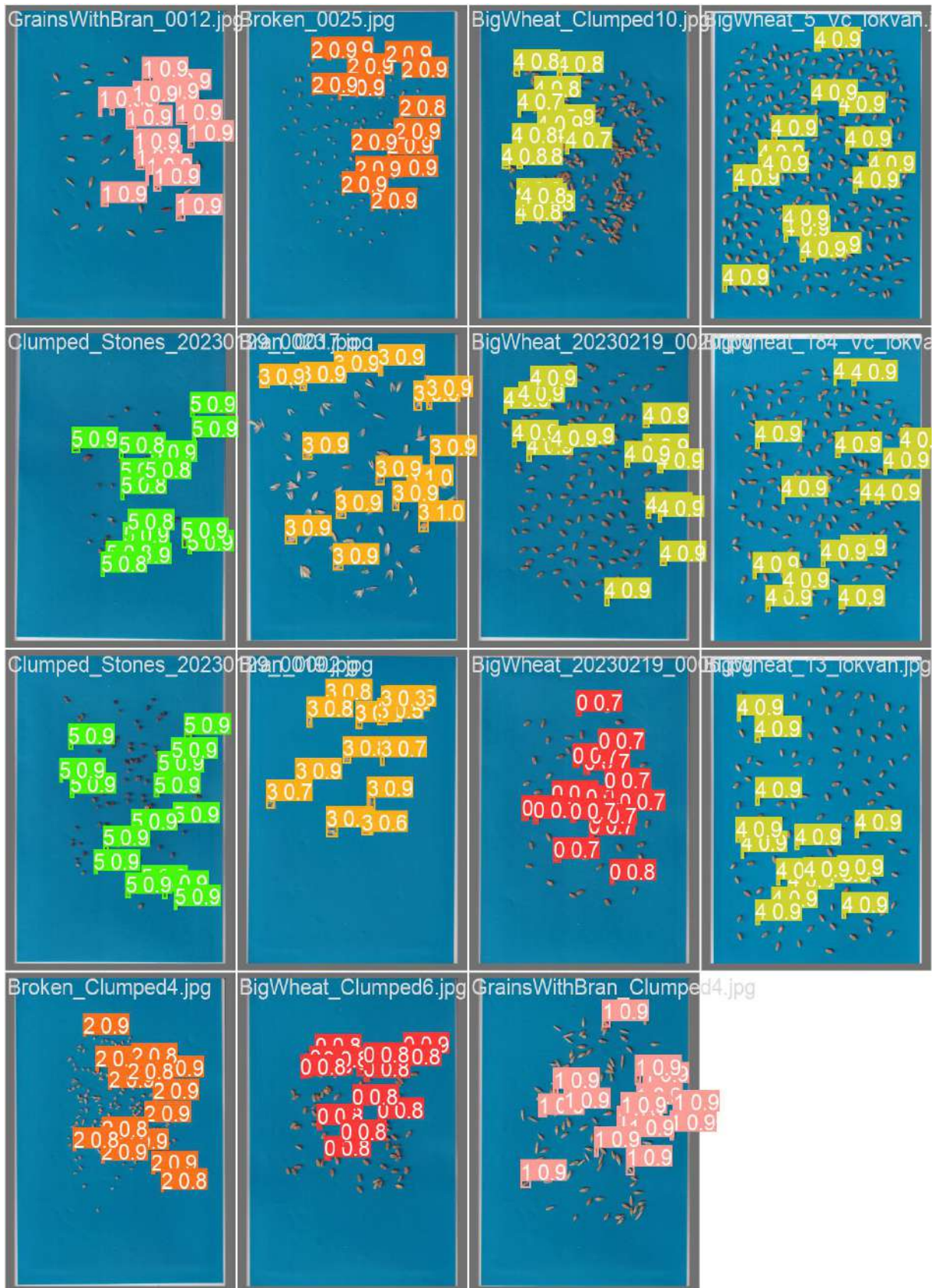
Test images batch1 with correct labels



Test images batch1 with model predictions



Test images batch2 with correct labels

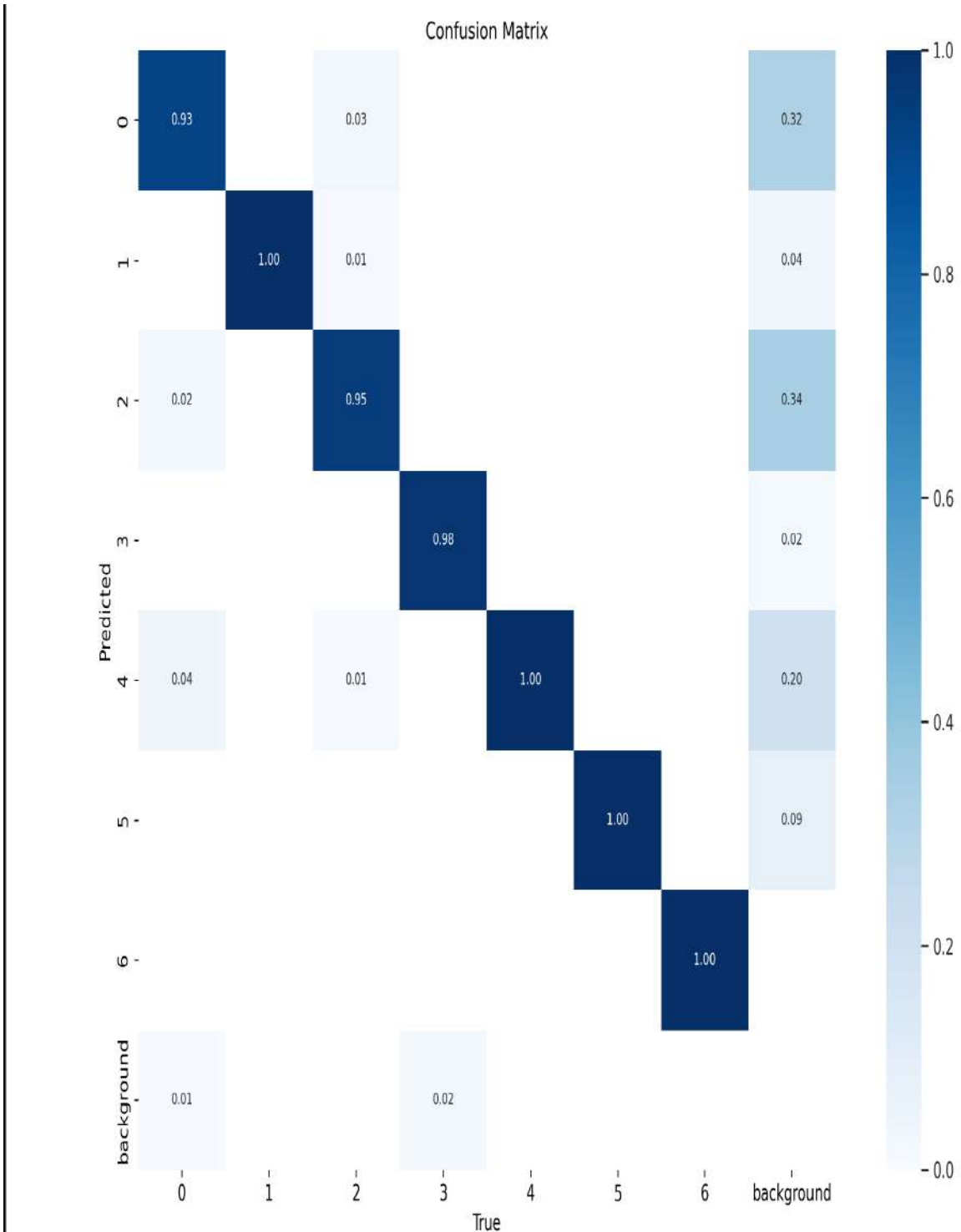


Test images batch2 with model predictions

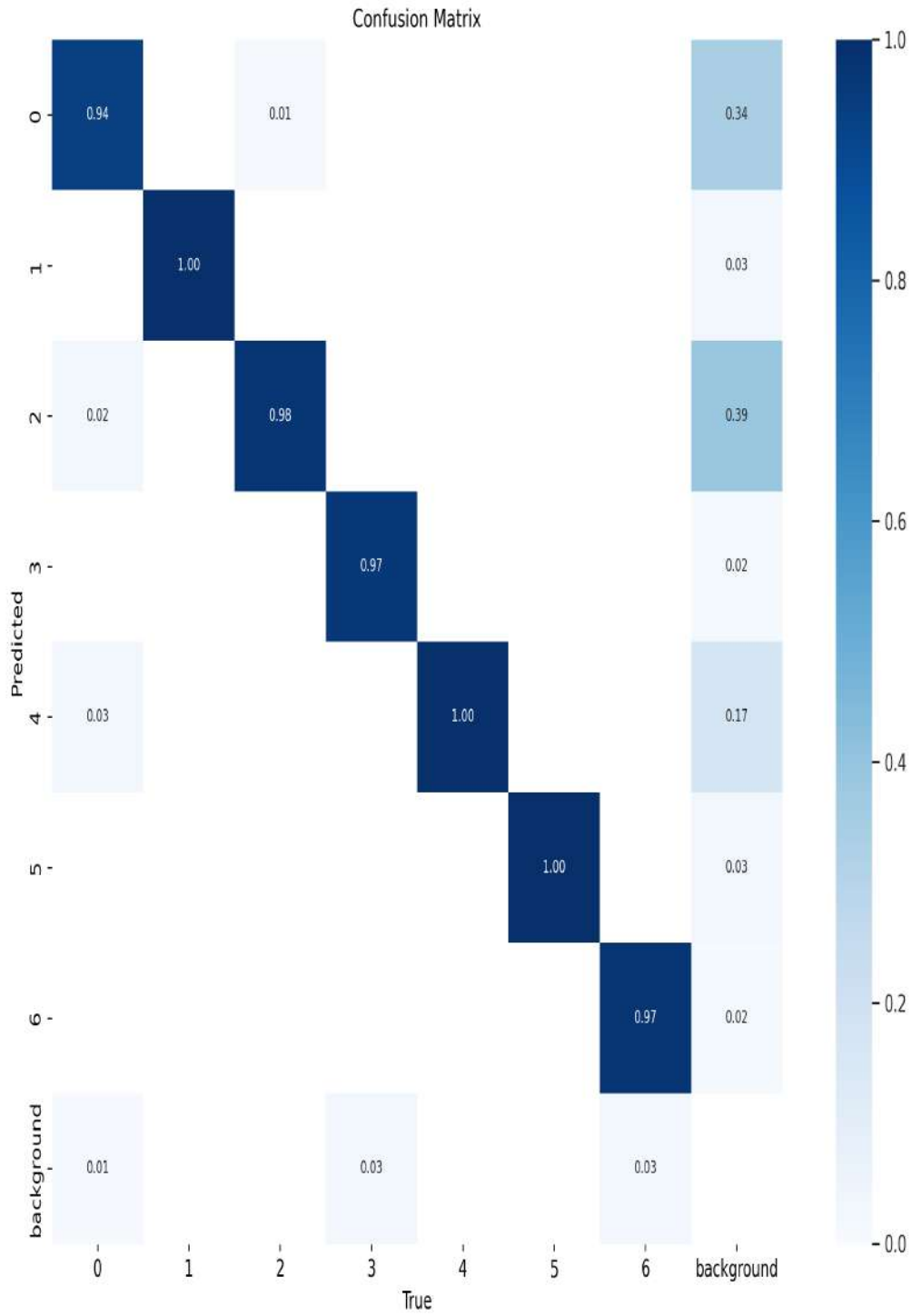
- The above images clearly show that performance of this model is not at all good. Hence, trying a model with bigger size was decided.

4.1.2 | Yolov8l: Large model

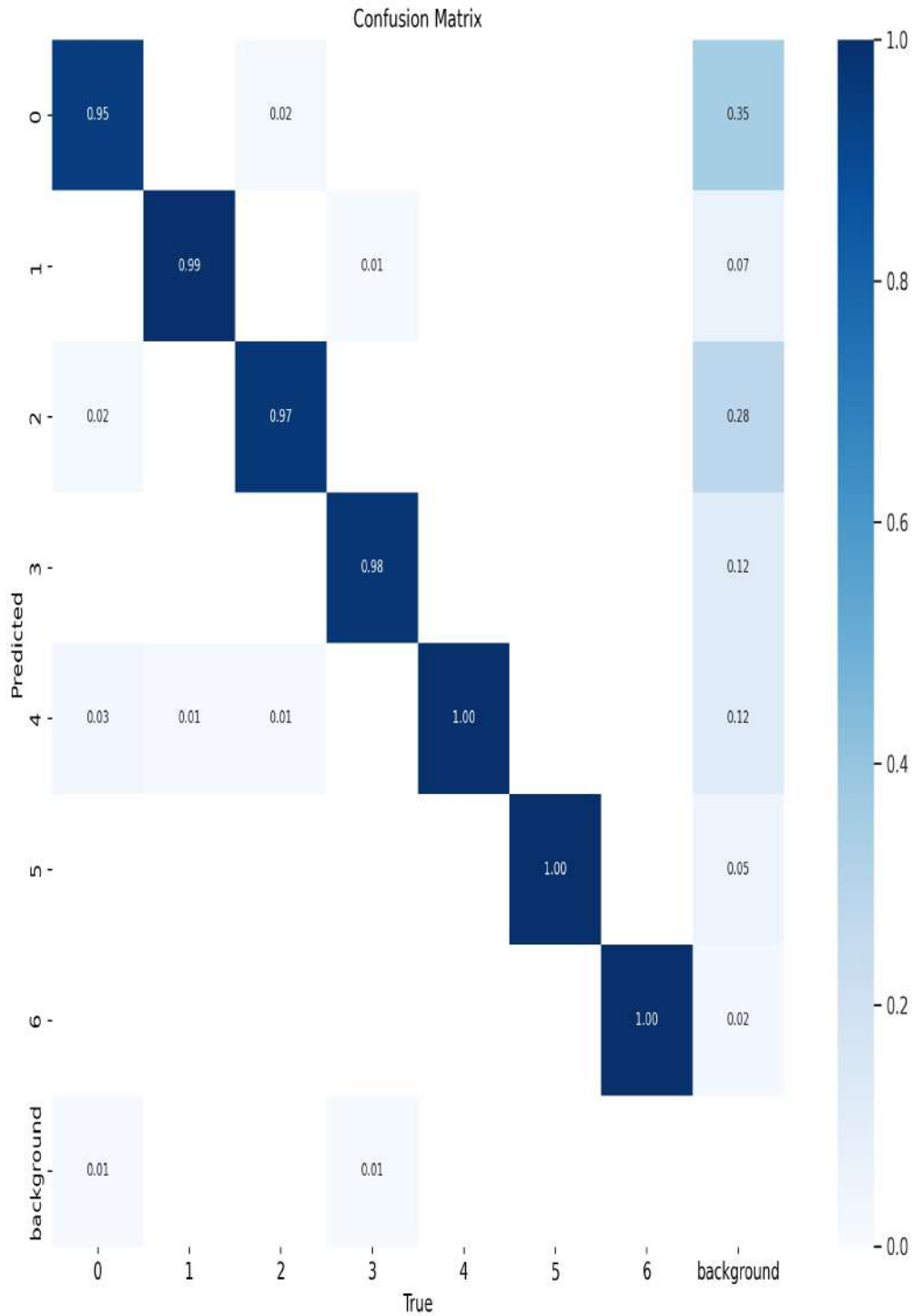
- Below are the confusion matrix images



After 40 epochs

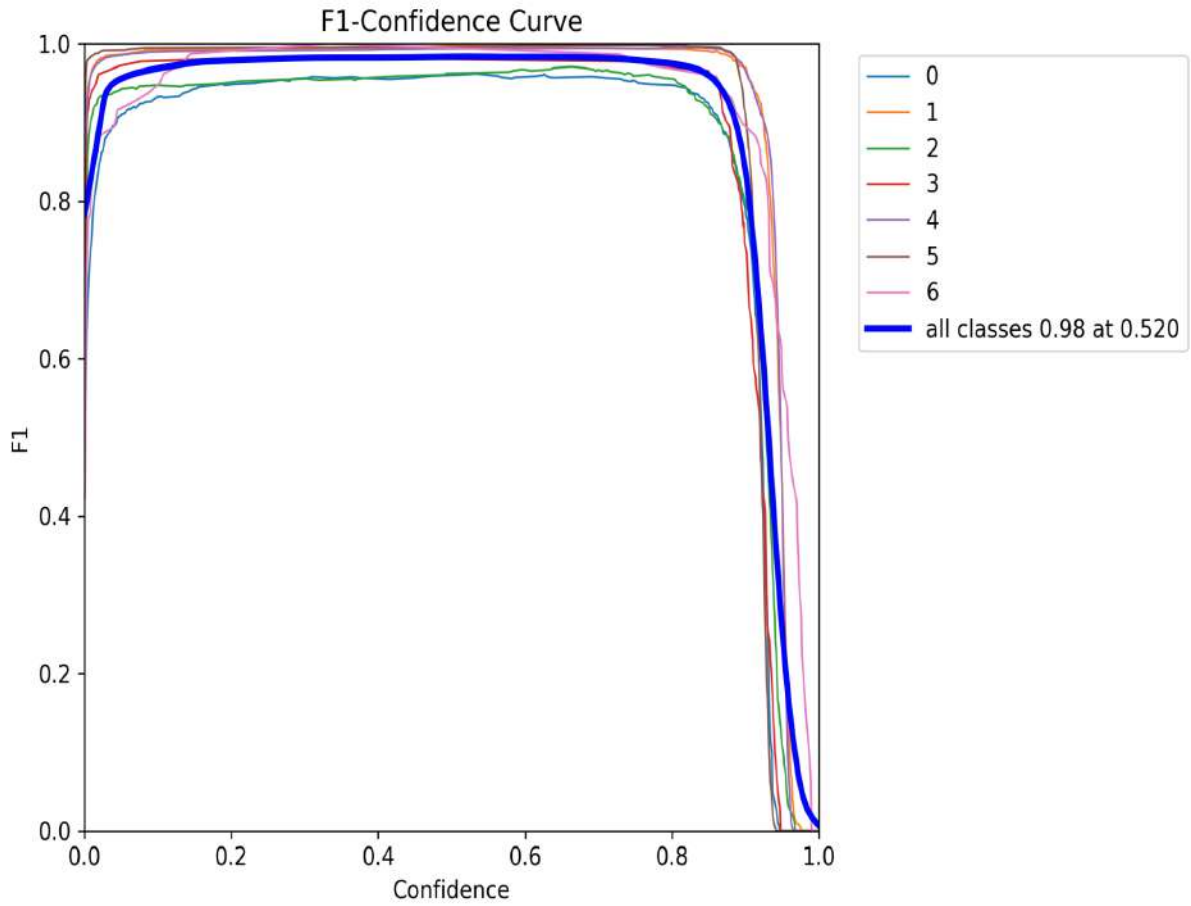


After 80 epochs

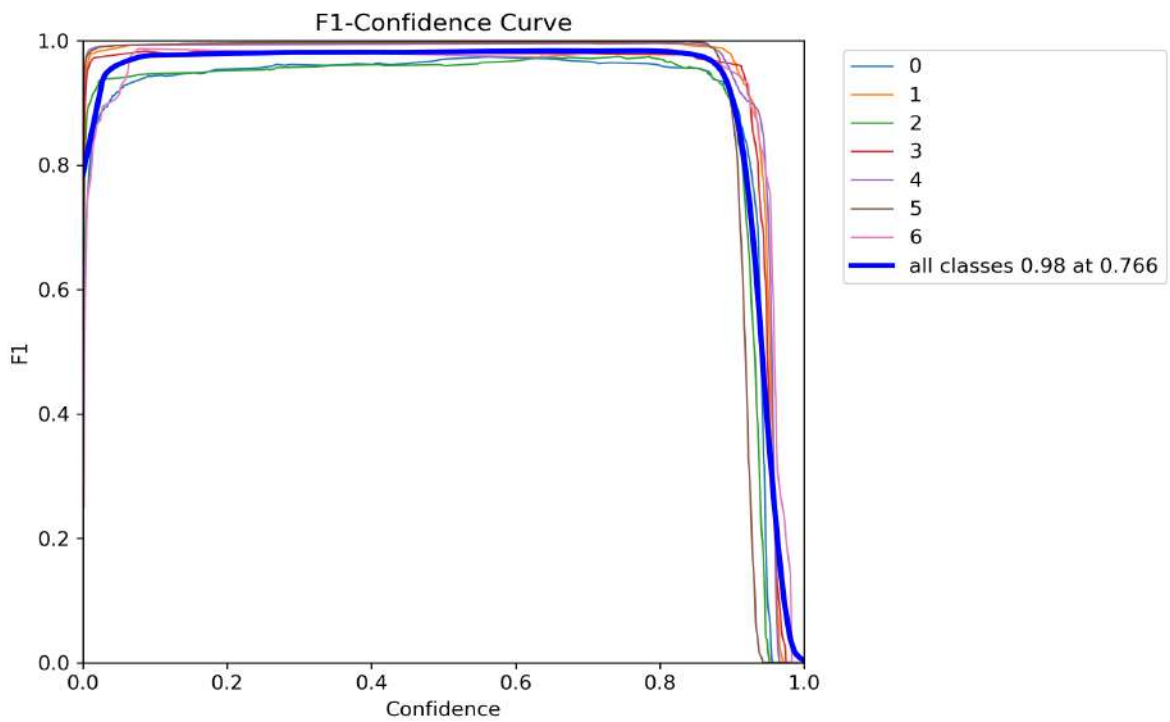


After 120 epochs

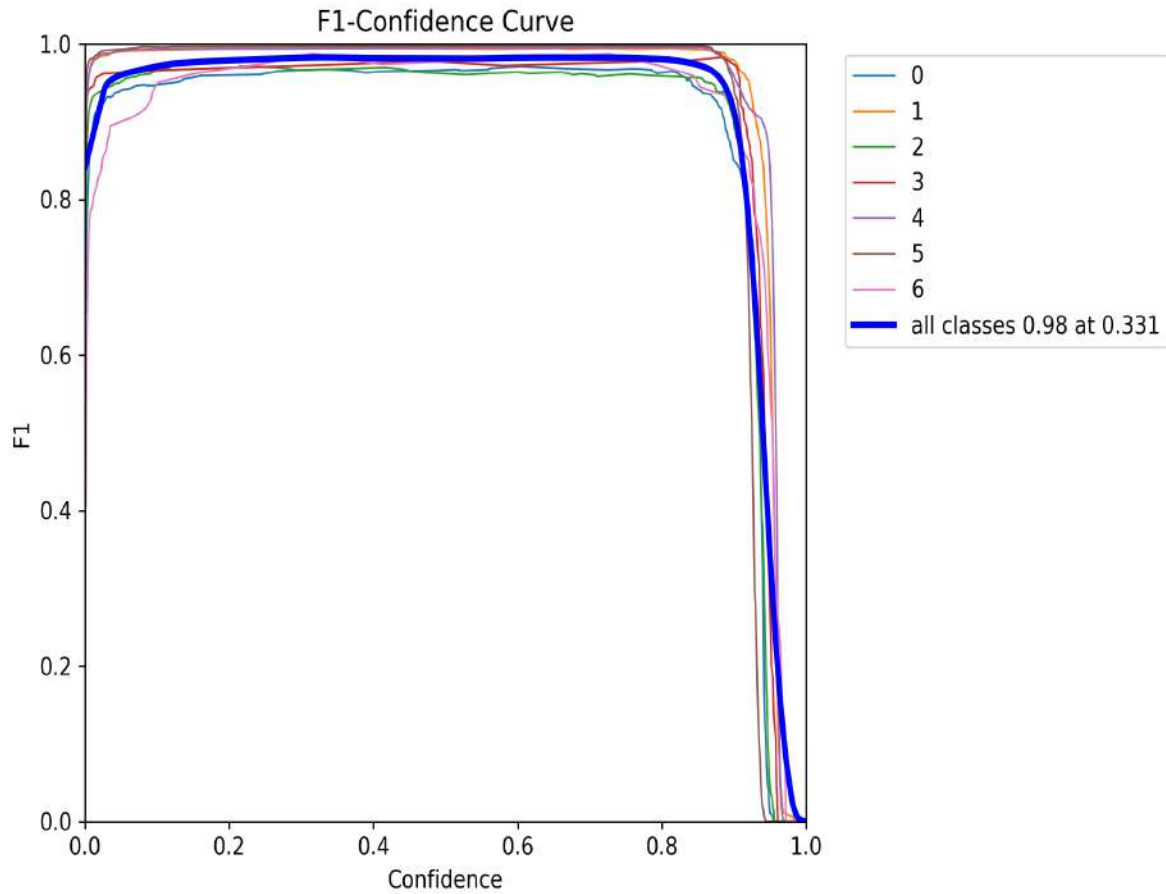
- As can be seen from the above images, the performance of model is increasing while we increase the epochs. After training for 120 epochs, the model predictions accuracy for almost all classes are close to 1.
- Below F1 curve images are shown.



F1 curve after 40 epochs

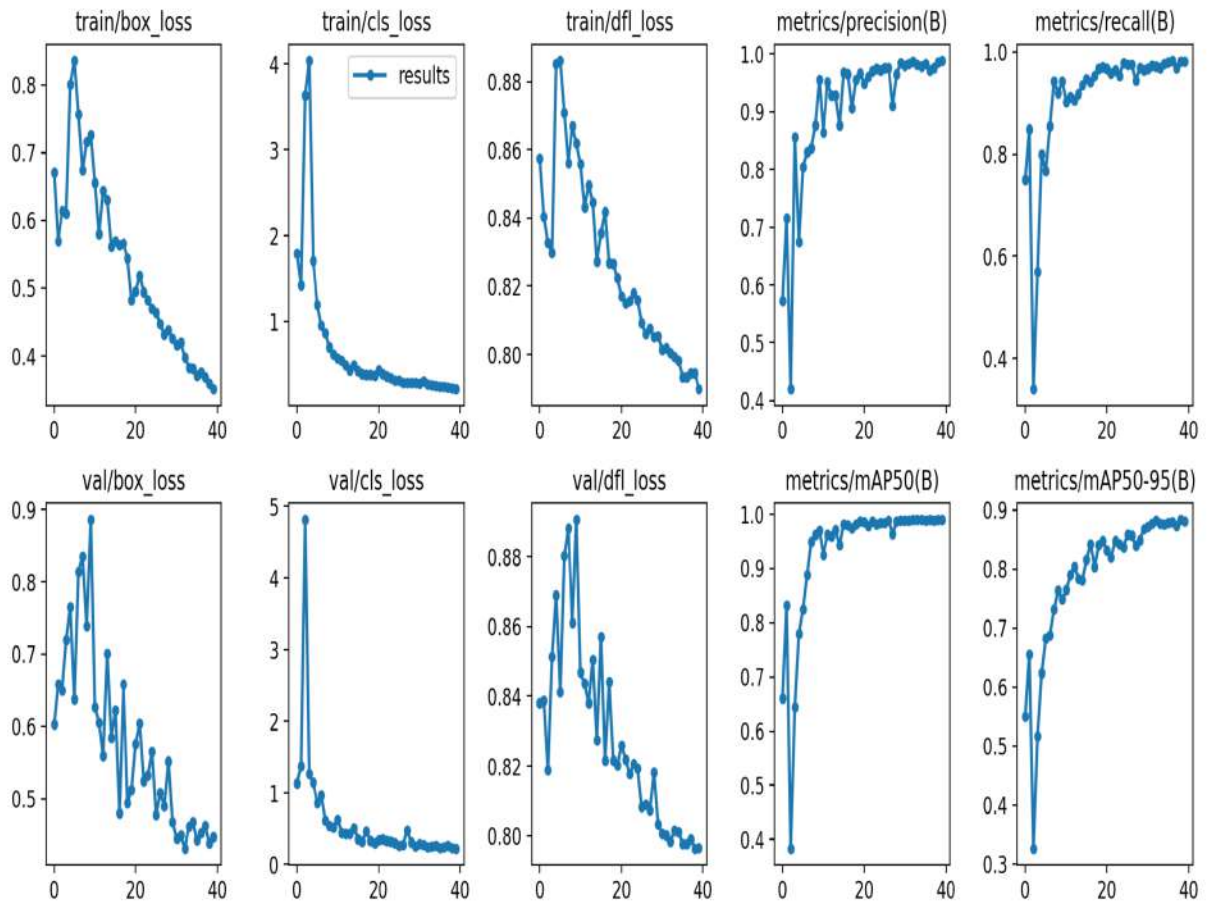


F1 curve after 80 epochs

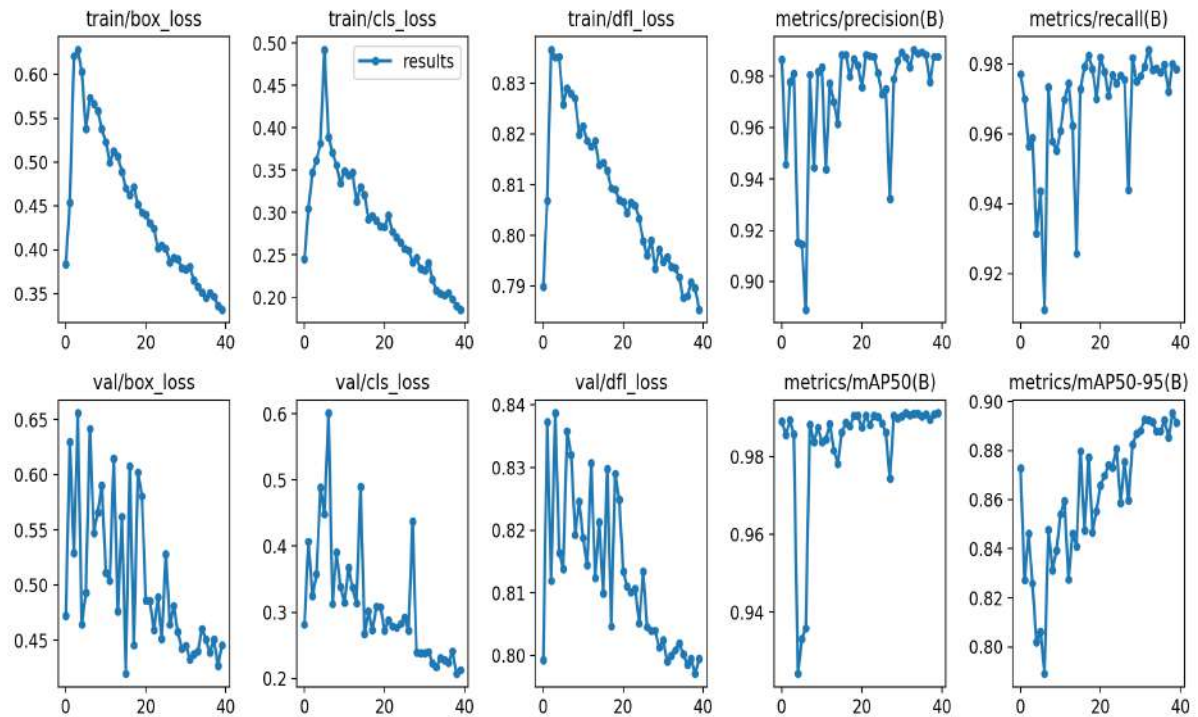


F1 curve after 120 epochs

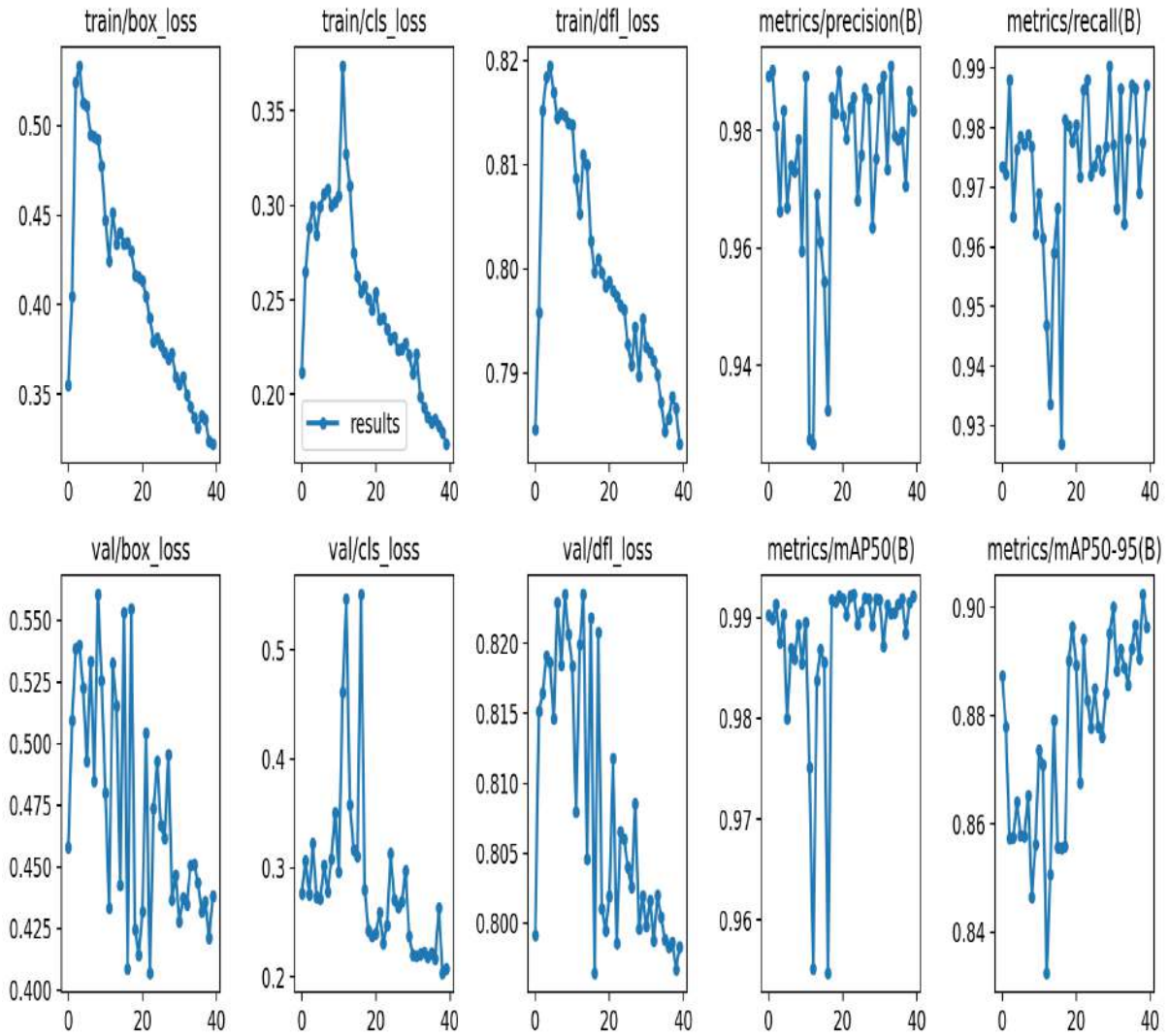
- As can be seen from the above images, the confidence required for 0.98 F1 is decreasing on increasing the training. After training for 120 epochs, the confidence required is minimum which is 0.331.
- Below are the images of results.



After 40 epochs

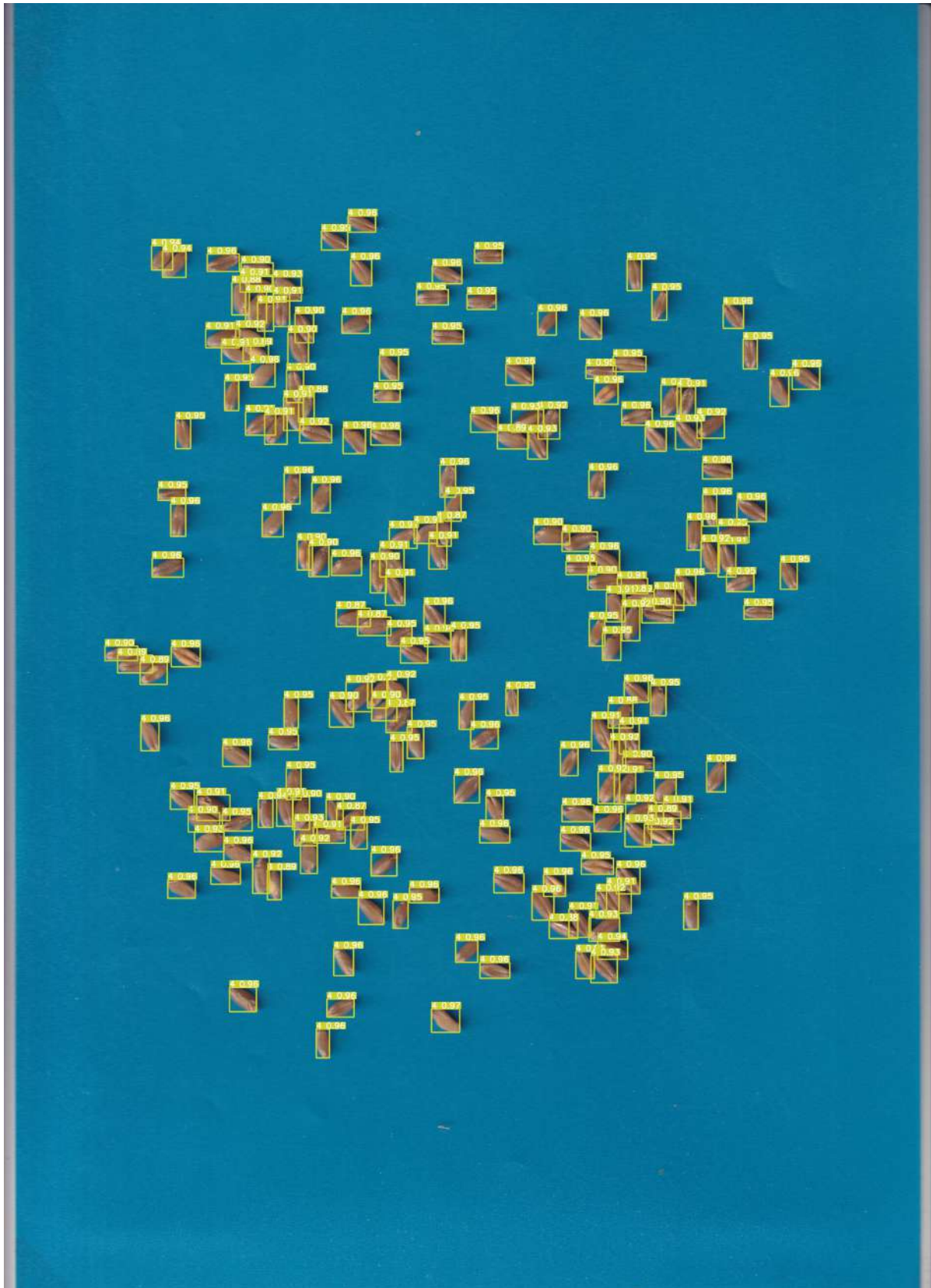


After 80 epochs



After 120 epochs

- Below are the images showing how the yolov8l model(trained for 120epochs) is performing on the test data images



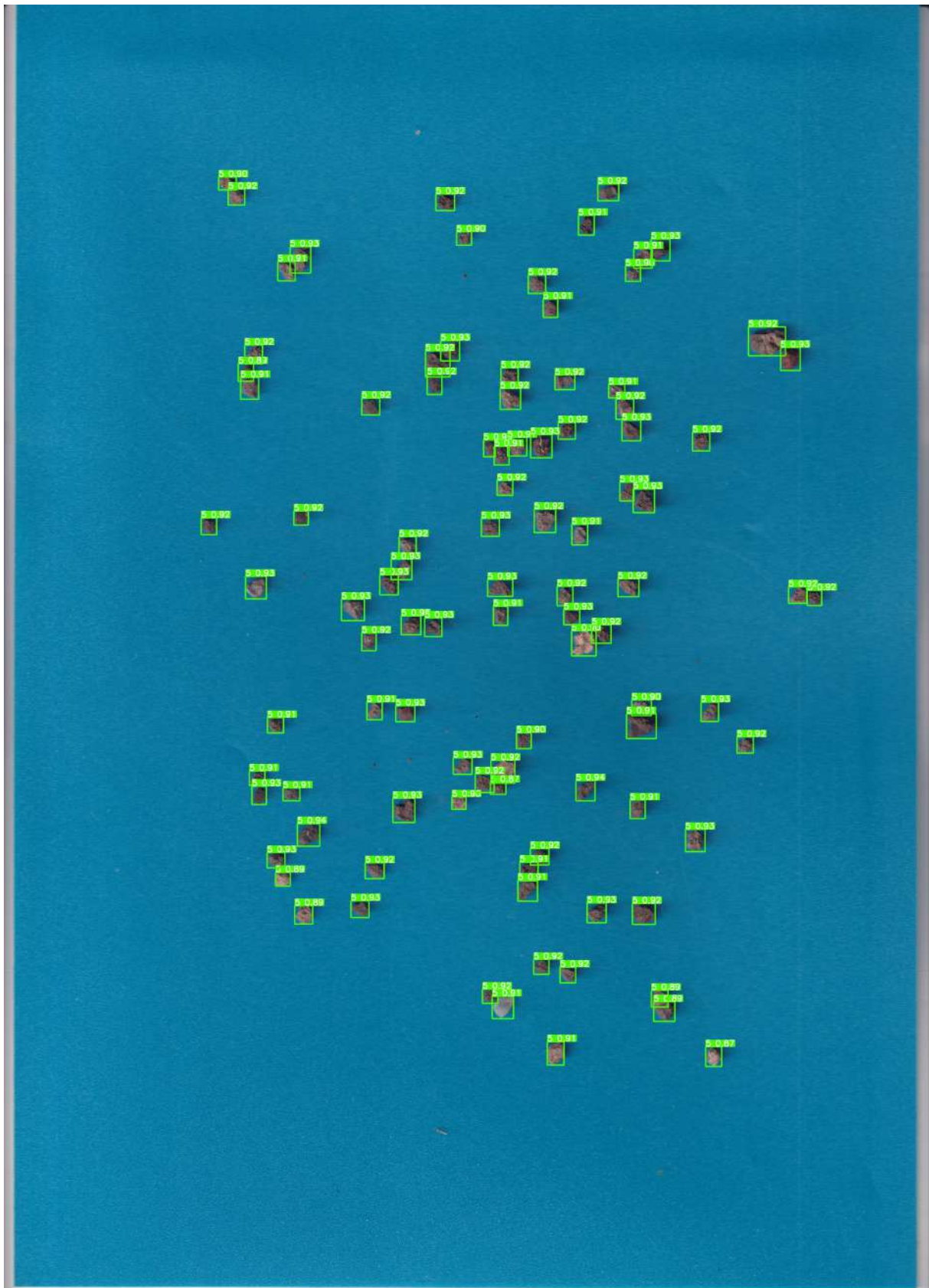
Big wheat clumped test image



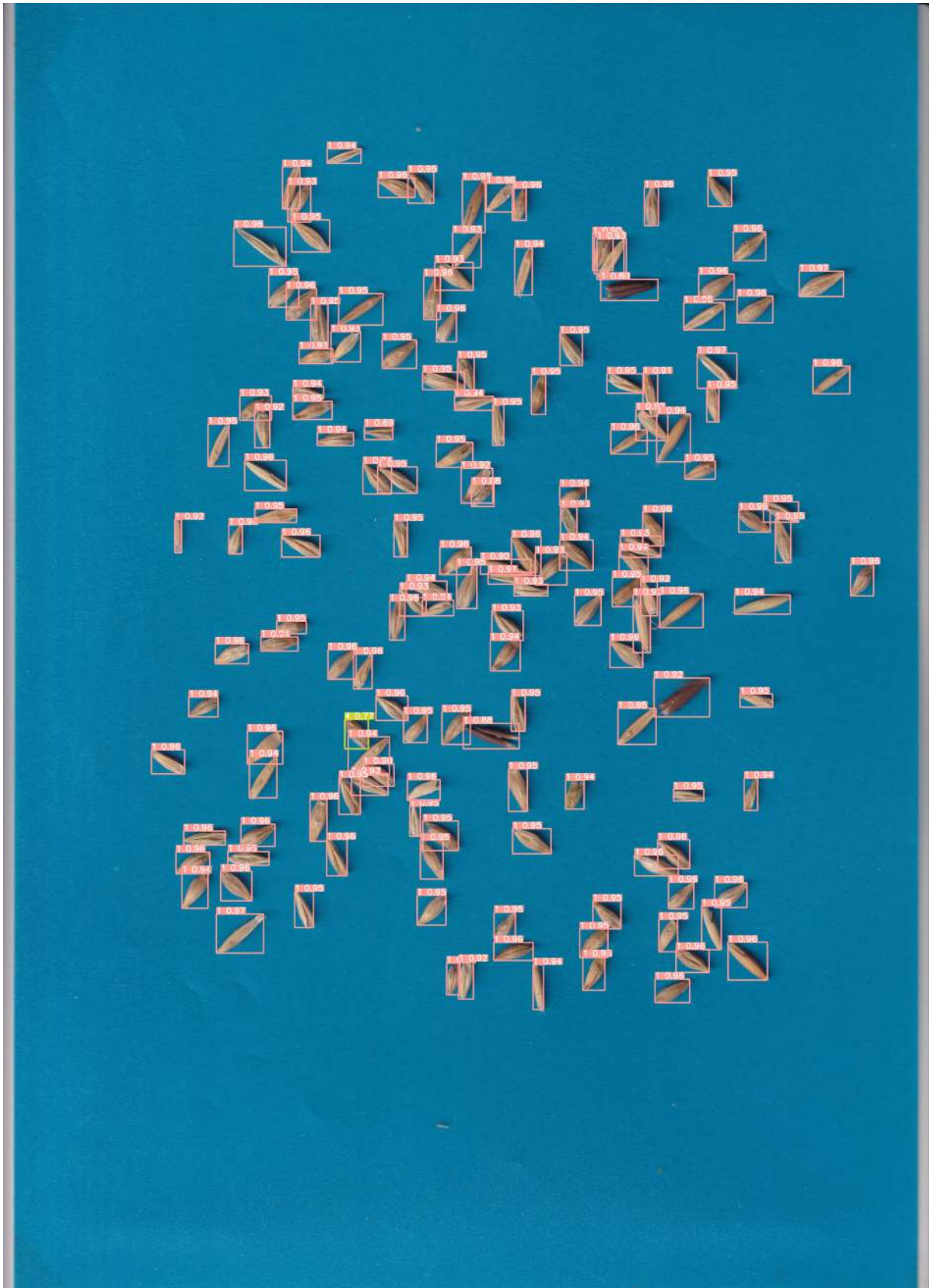
Bran test image



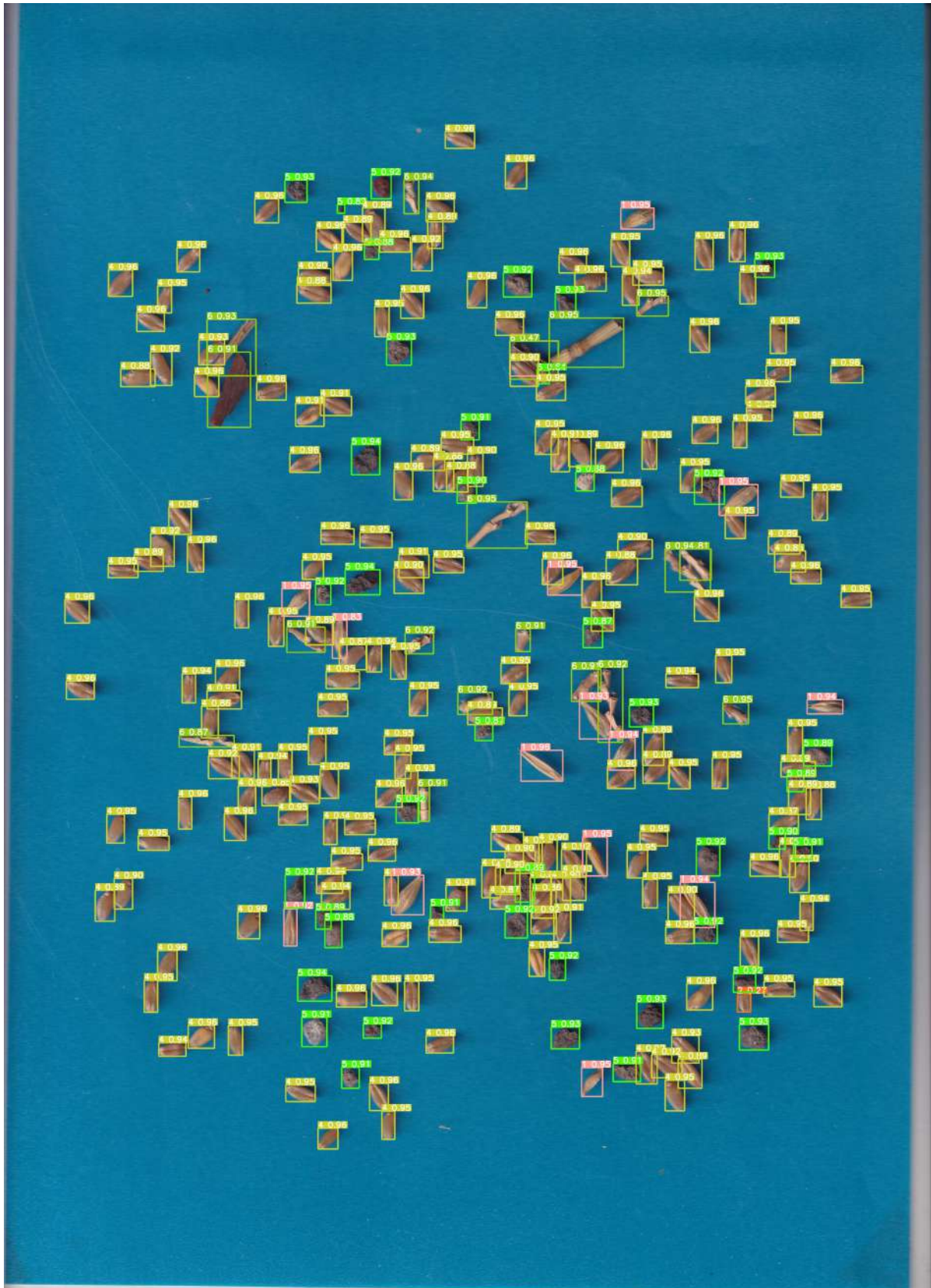
Clumped broken grains test image



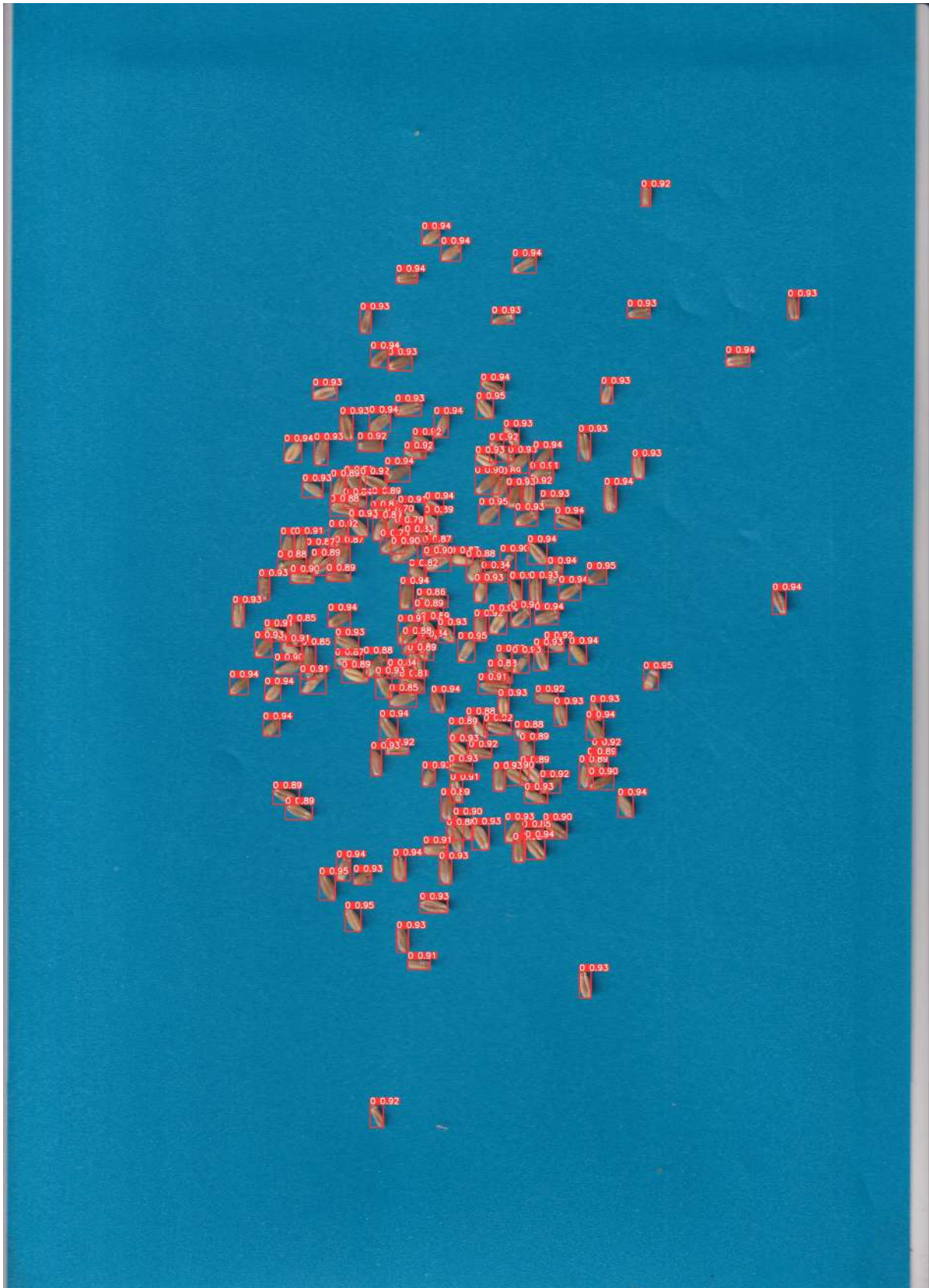
After 120 epochs



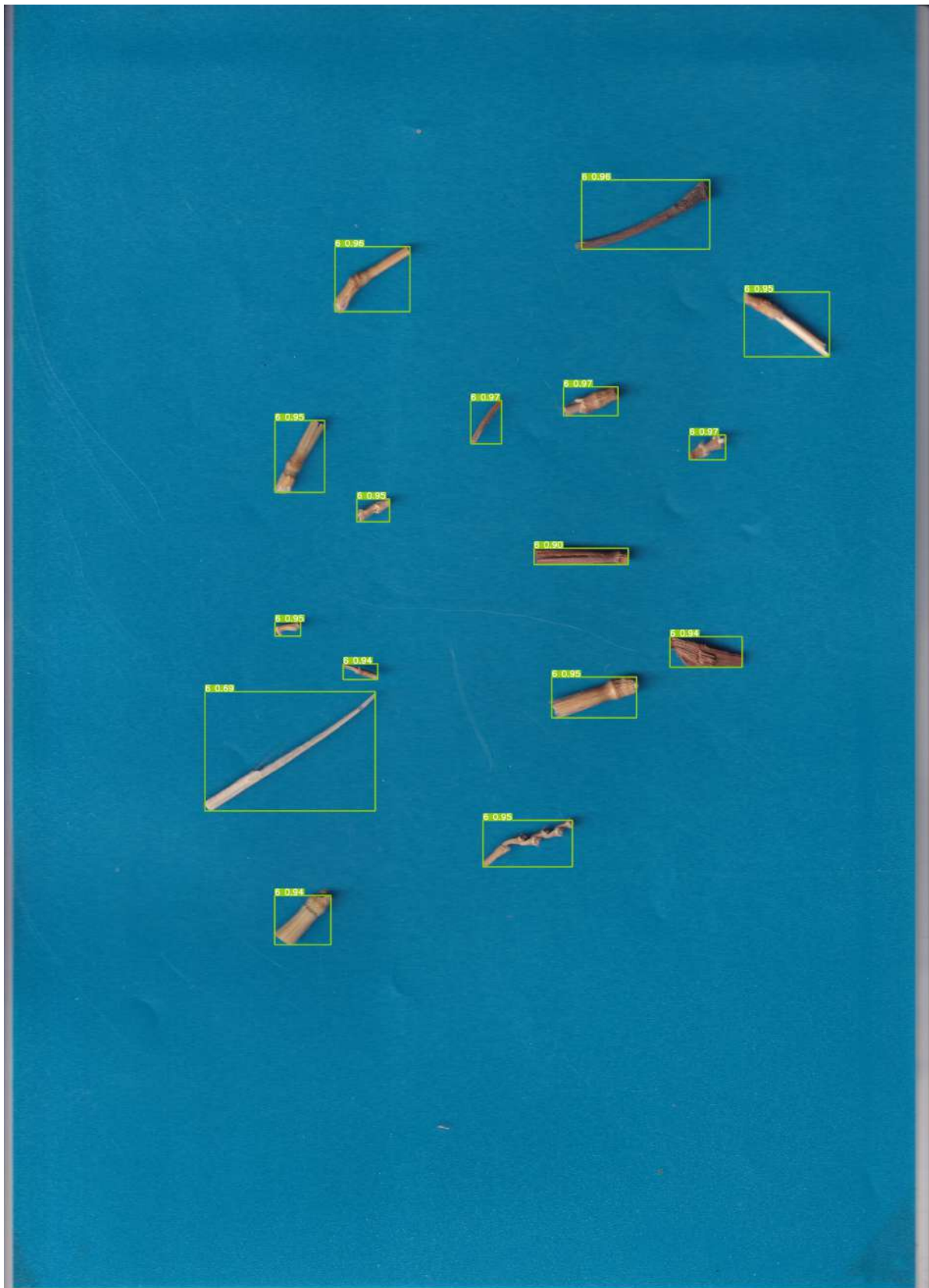
Clumped grains with bran test images



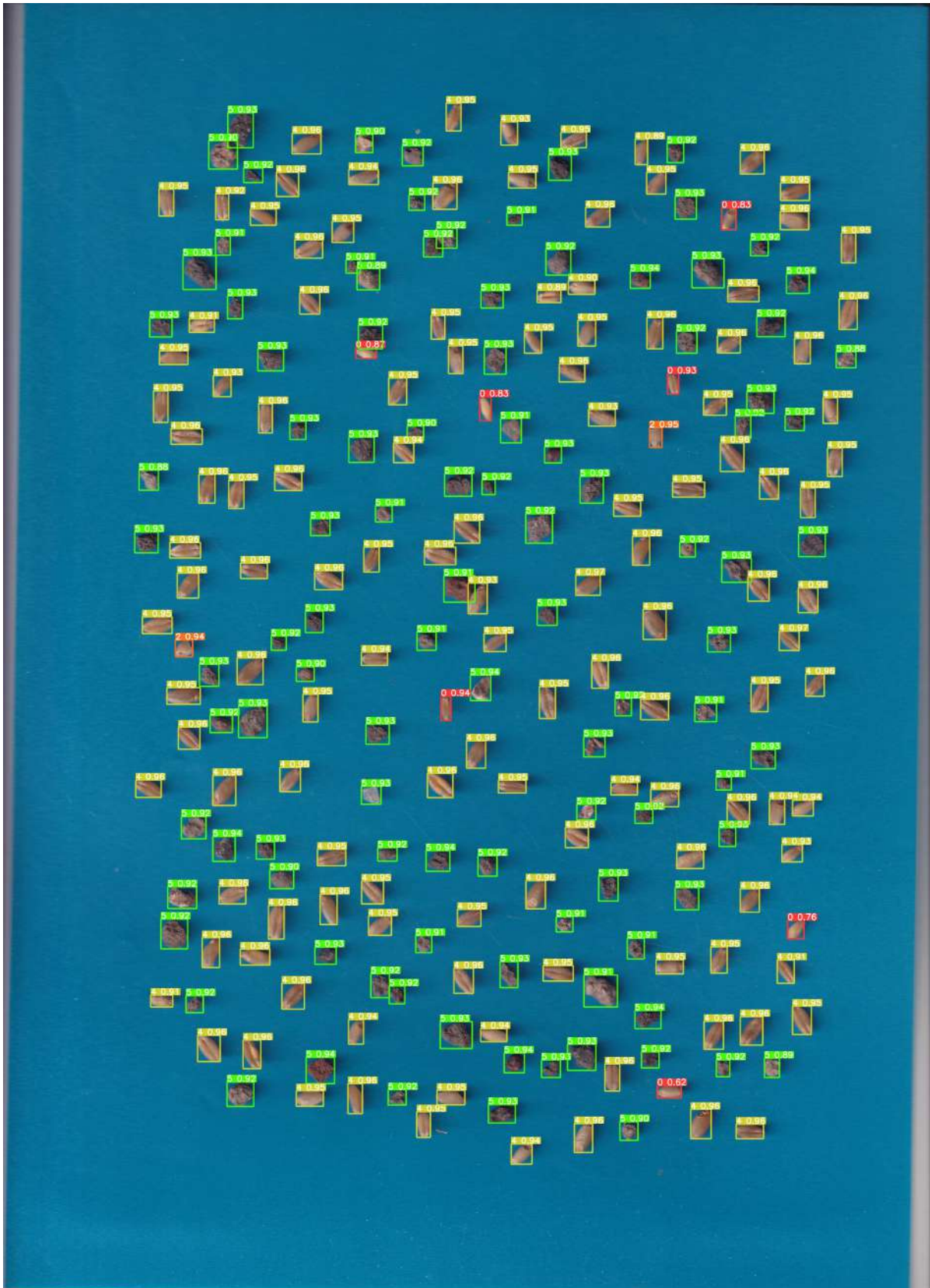
Mix (Combination3) clumped test images



Clumped small wheat grains test images



Stalk test images



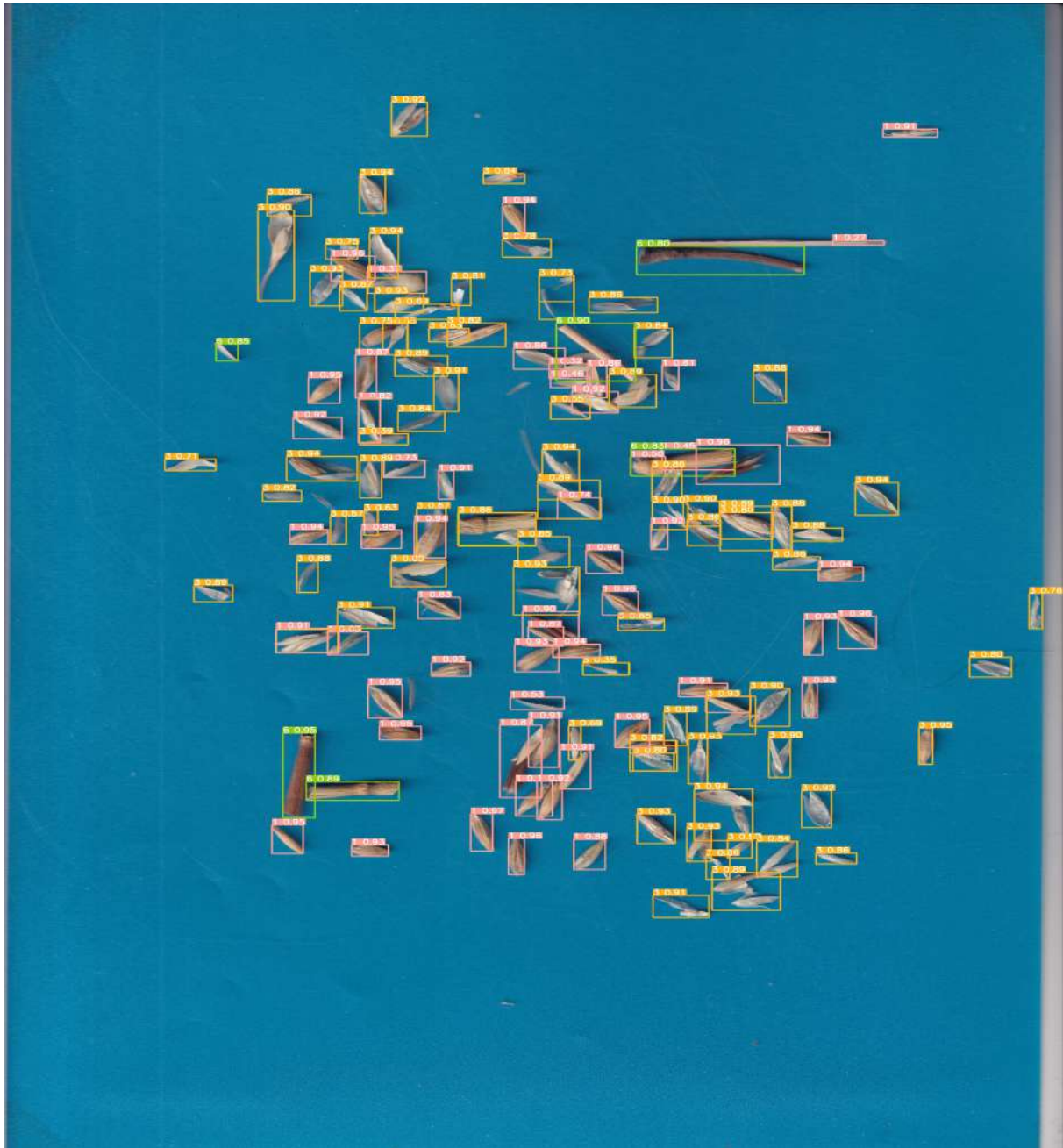
Stones and grains (Combination2) test images

- The above images shows that model is performing really well on predicting every class of objects. Also objects in clumps are being detected individually.

[Model predictions on all 31 images of test data is here.](#)

4.1.3 | Model predictions on new images taken.

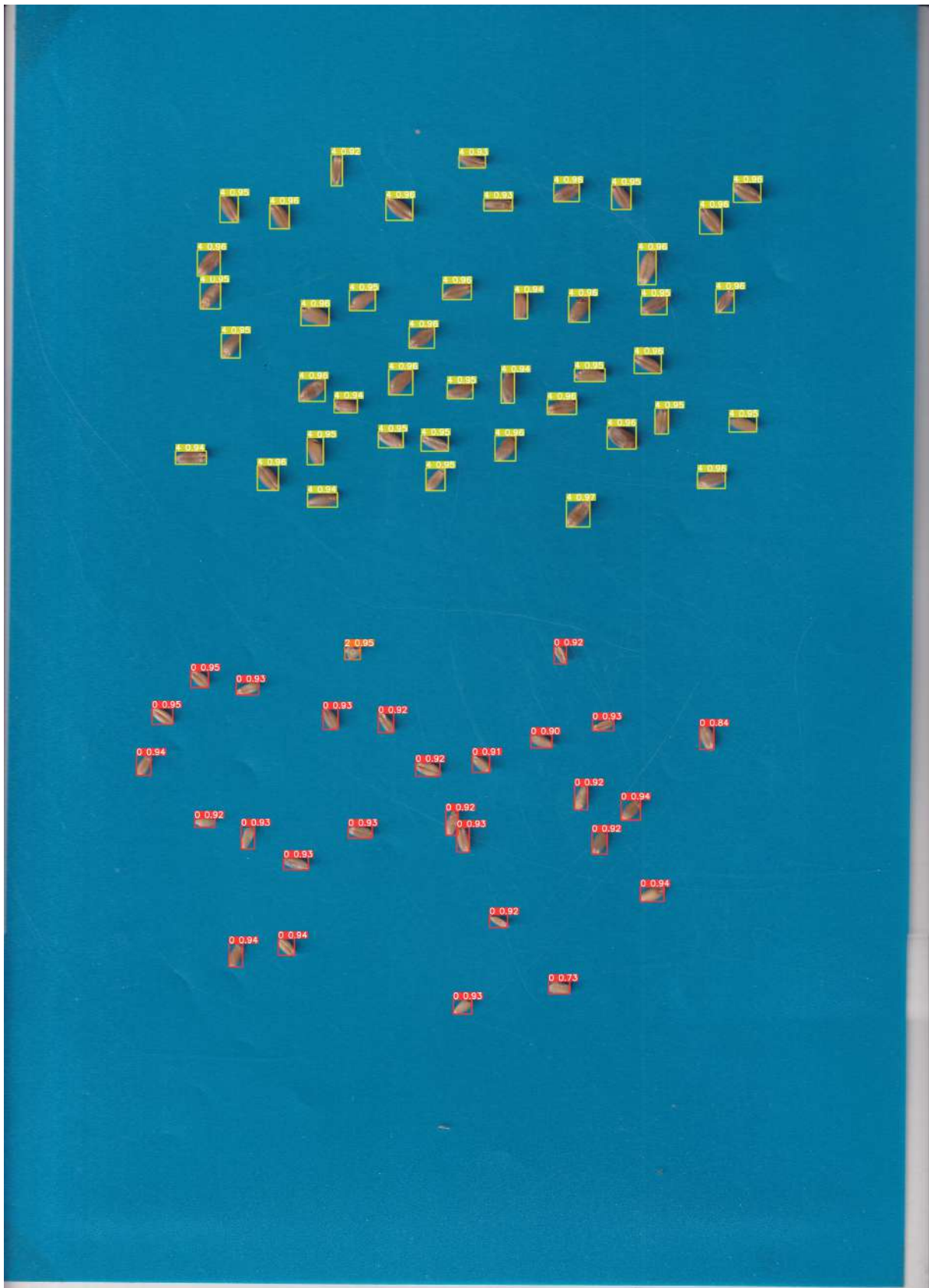
- New scans are taken to test after the model was performing good on test data. From the samples collected, took scans of 1) Combination 1:- Grains with Bran + bran + stalk 2)Combination2:- (Mix) : big wheat + stones + wheat brans + broken wheat grains + stalk 3)Combination3:- small wheat grains + big wheat grains 4)Small wheat grains only.



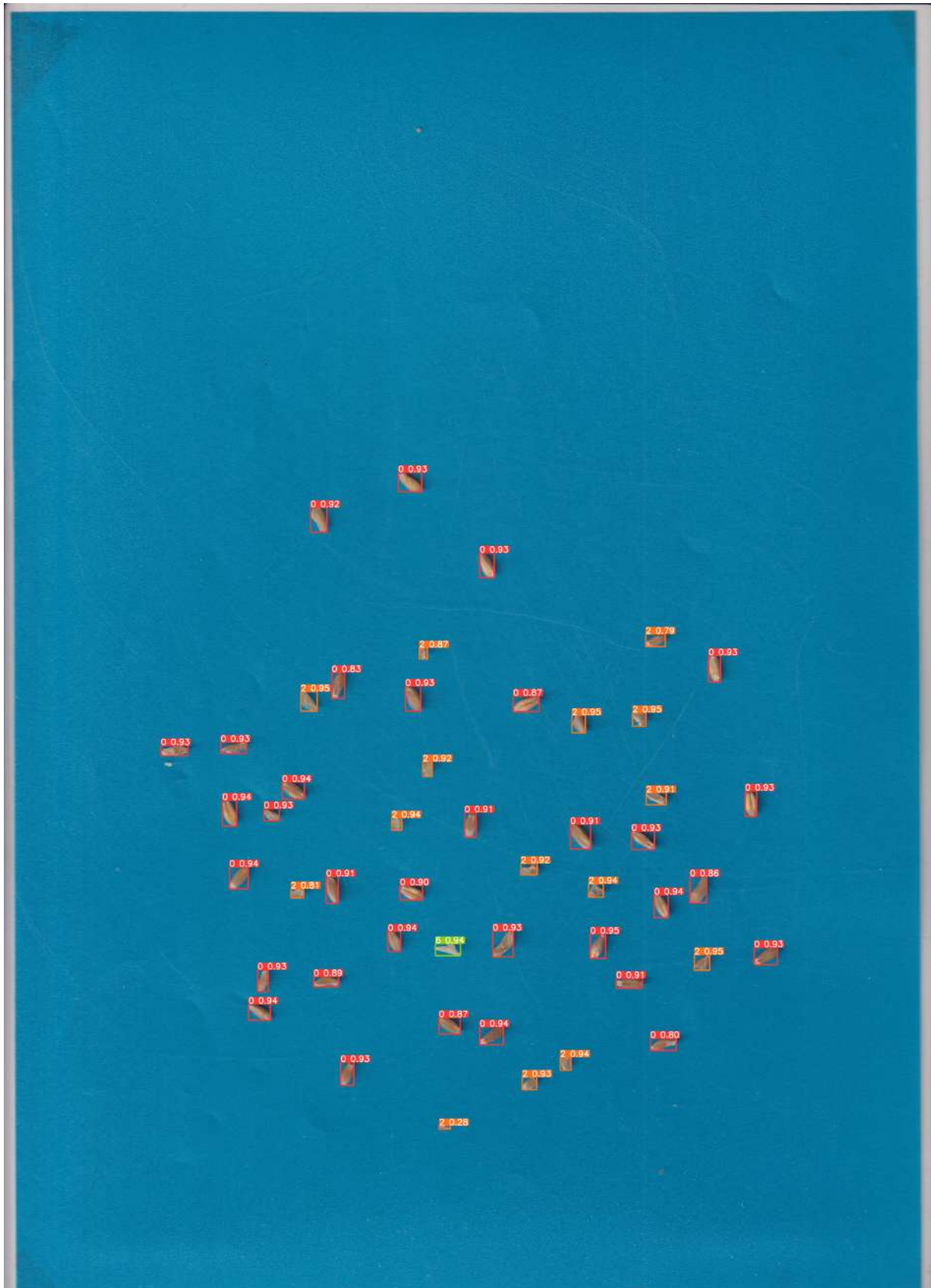
Grains with bran + bran + stalk



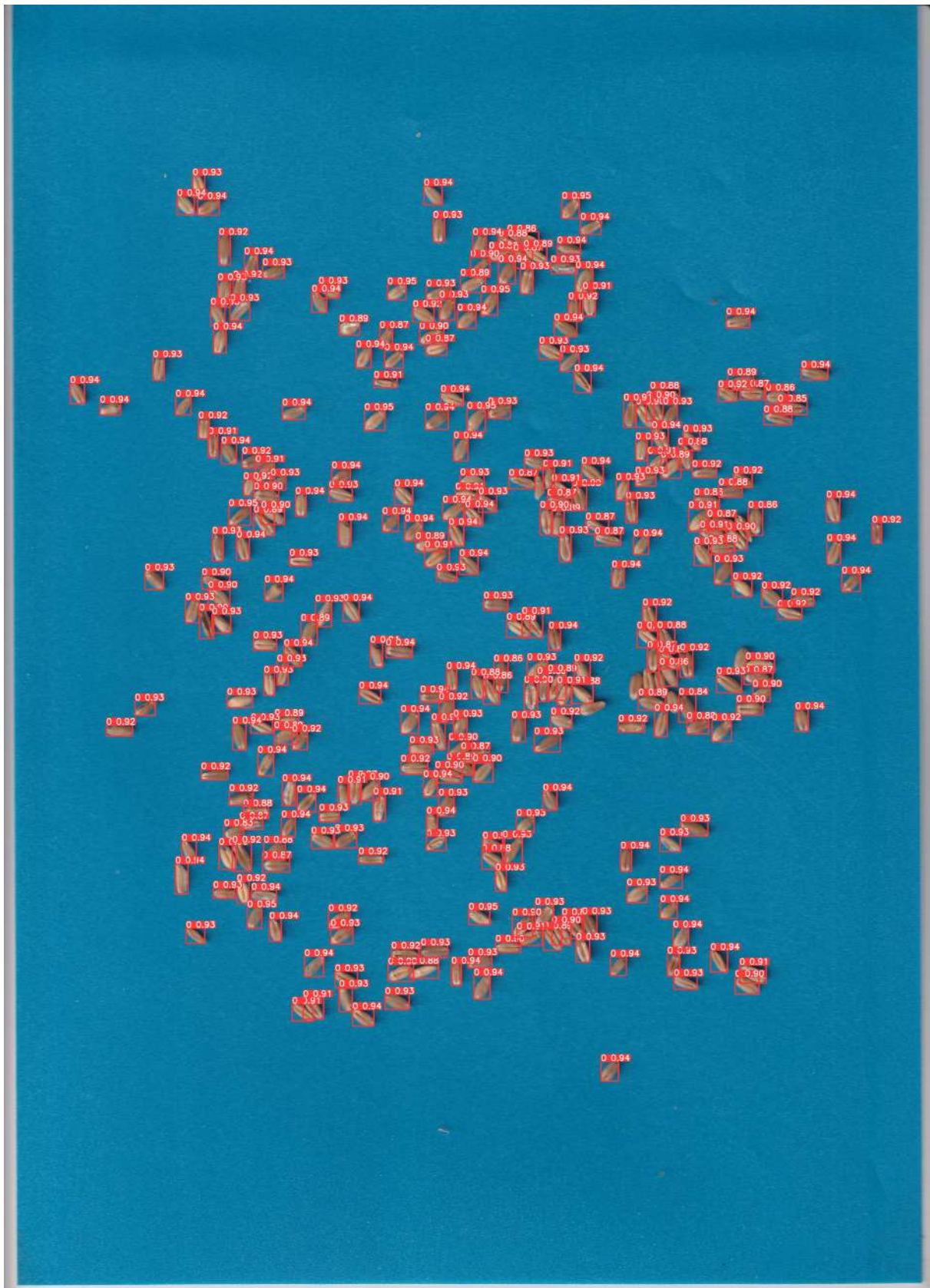
Mix



Small wheat grains (below) + Big wheat grains (above)



Small and broken wheat grains



Clumped Small wheat images

- The above images show that model is working very well. Also it has learned the difference between small grains and big wheat grains which is sometimes also difficult

to distinguish with naked eye.

- Below are some images in which the blue background is removed. This is just to see how the model works without blue background.



grains with bran + bran + stalk (without background)



Mix (without background)

- As can be seen from above images, the performance of the model is not as good as that in images with background but it is still mostly accurate in predicting the class.

4.1.4 | Results on the Dashboard

- In our user dashboard, user has to upload the scanned image and table of counts of each type of class will be displayed with the image showing predicted classes by the model. Some images from test data is uploaded to the dashboard and the results are shown in the below images.

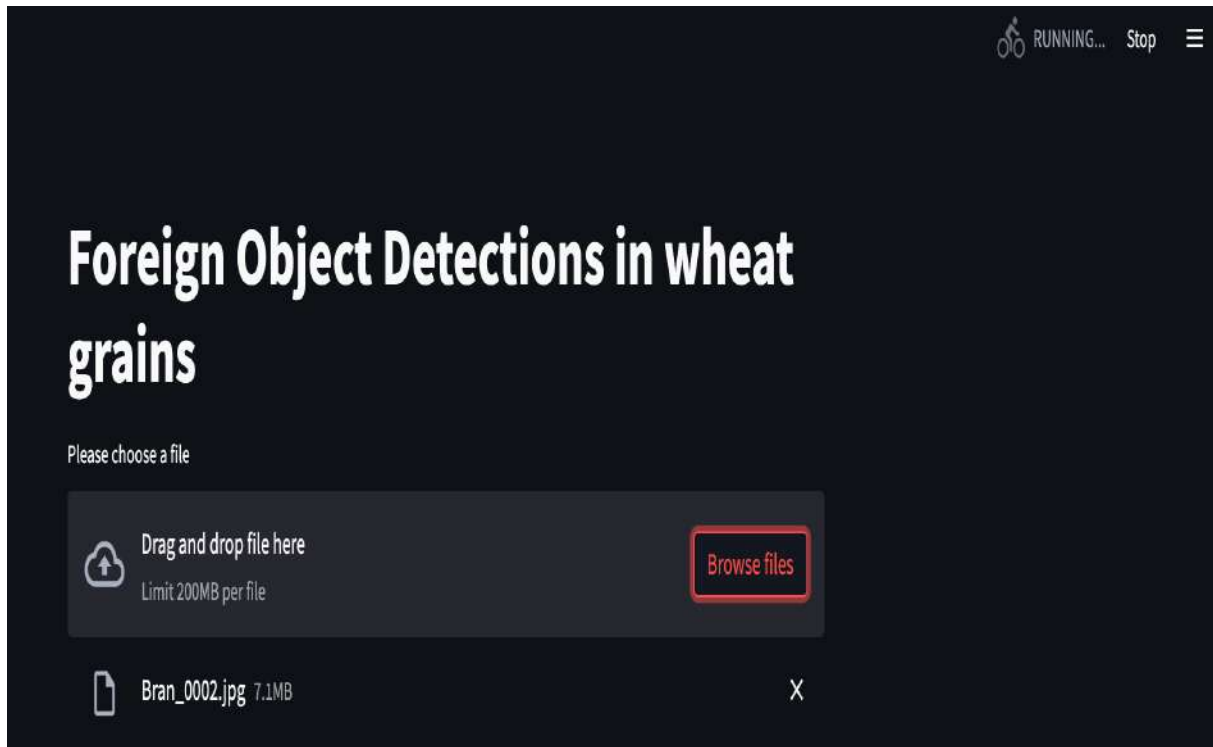


Image is uploaded and model is processing it

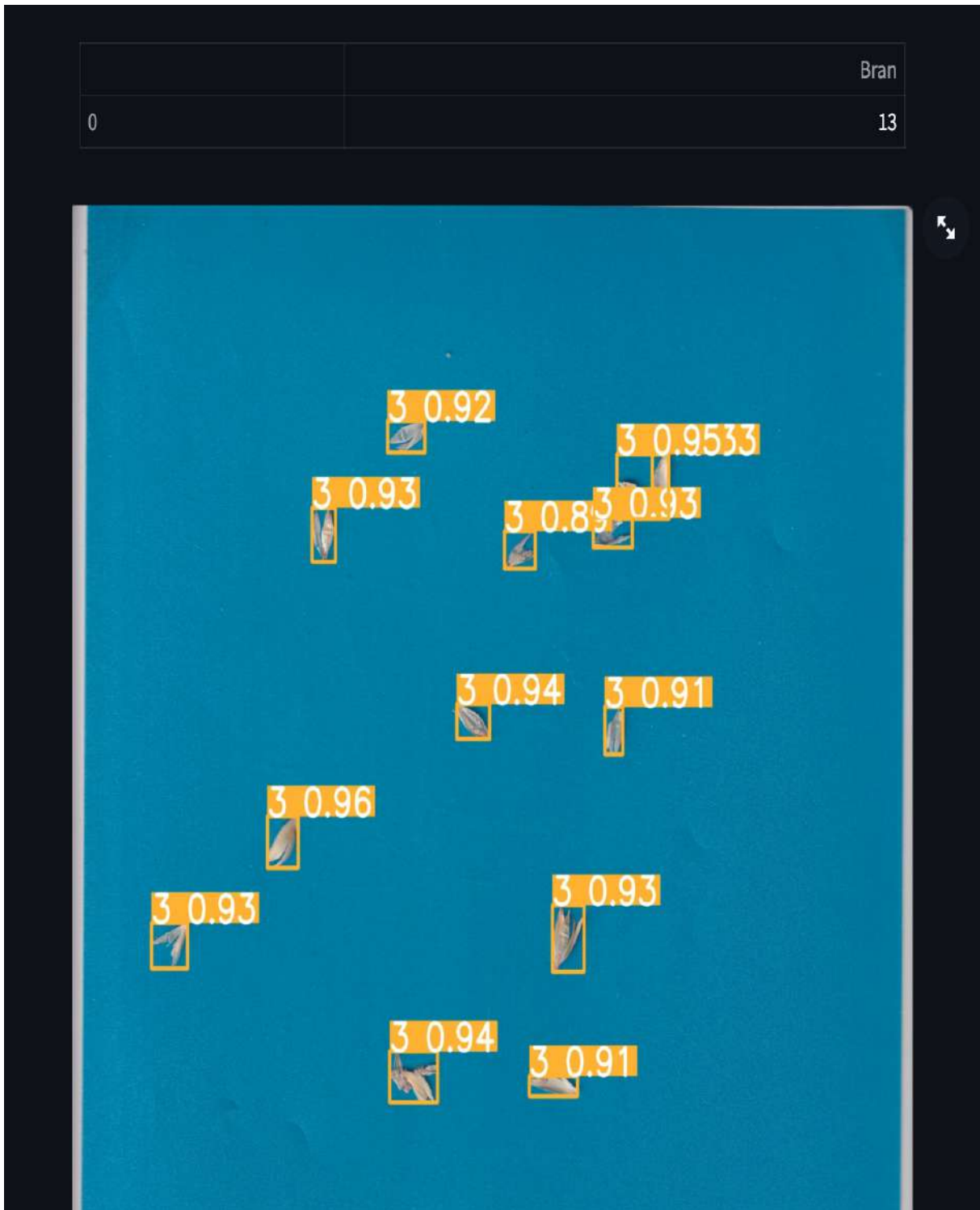


Table showing count of objects

Drag and drop file here
Limit 200MB per file Browse files

Mix_0005.jpg 4.4MB ×

	Grains With Bran	Big Wheat Grains	Stones	Stalk
0	26	36	8	4

Table showing counts of objects



Image with all the predictions of the model

4.2 | Project Outcomes

My project provides dashboard to user where he/she can upload the scanned image and get the analysis with predictions done by model shown in the uploaded image. This tool is more reliable because it provides objective analysis which is better than relying on one's own subjective analysis.

4.3 | My Contributions to the Project

The whole project from scratch was done by me with guidance of my mentor Sanket Patel sir. From field visit for collecting the samples to then collecting and annotating the data to then training the model and at last creating the dashboard for user, whole process was done by me.

4.4 | Learning Outcomes

I decided to do this project because it involves the whole process of a computer vision engineer from finding ways to collect the data, then collecting proper data, then selecting the right model for the problem, then training the model, then check the results and at last creating a dashboard where the model can be accessed by layman. I learned all these things in this project. Most importantly, I learned how data is very crucial for building a great model. I learned that understanding the problem, and collecting the right kind of data accordingly is very crucial.

4.5 | Real World Applications

My project can serve as a base for building a software which can be commercialized. In my project, I have created a basic dashboard which can be used in real world but it requires more effort from user to first take the scan and then upload the image on dashboard to get the analysis. If the user wants to share the analysis then he needs to share it with some other communication apps. The future work would involve building a software or web app on top of my project, which automatically takes the image from the scanner and it automatically uploads it on the dashboard and gives the analysis. This software or web app would also include feature of creating groups on which the analysis automatically gets shared to others in the group. This feature would help traders get the analysis from his various team members.

Chapter 5

Conclusion

This project started with first the field visit to understand the problem and learn about the different types of foreign objects that is found in wheat farm produce. After collecting the samples of foreign objects, the crucial stage of collecting the right kind of data in right manner was implemented. This project also involved literature survey which required me to learn to read and understand complex research paper to find any similar work that I was doing in the project. Literature survey was also very helpful to find the right kind of model that can be used for our problem. After the collection of data, annotation was the big challenge which required lot of time. Tackling this issue forced me to find smart way of using contours to get some initial annotations which would save time. Then YOLOv8 models of different sizes were trained to find which one works well. To test the trained model, new scanned images was also taken to see if the model has overfitted or not. To make this model accessible to any layman and to provide the analysis, dashboard was also created in this project.

Bibliography

- [1] C. Z. B. L. . J. W. . G. Yin Shen, Yanxin Yin, “Image recognition method based on an improved convolutional neural network to detect impurities in wheat,” *IEEE Access*, 2019.
- [2] A. Mehra, “Understanding yolov8 architecture, applications features.” <https://www.labellerr.com/blog/understanding-yolov8-architecture-applications-features/>, 2023.
- [3] F. Jacob Solawetz, “What is YOLOv8? The Ultimate Guide.” <https://blog.roboflow.com/whats-new-in-yolov8/>, 2023.
- [4] R. Kundu, “Understanding yolov8 architecture, applications features.” <https://www.v7labs.com/blog/yolo-object-detection>, 2023.
- [5] P. Lebedzinski, “Understanding yolov8 architecture, applications features.” <https://towardsdatascience.com/a-single-number-metric-for-evaluating-object-detection-models-c97f4a98616d>, 2021.